VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Automatizované generování modelů v systému Simulink dle výsledků výpočtů v prostředí MATLAB

Automated Simulink Model Generation from MATLAB Design

2016 Miroslav Pazdera

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Miroslav Pazdera**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Automatizované generování modelů v systému Simulink dle výsledků výpočtů v prostředí MATLAB
Automated Simulink Model Generation from MATLAB Design

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je popsat strukturu "mdl" souboru systému Simulink a dále popsat možnosti jeho automatického generování pro známé struktury modelů dle výsledků výpočtů v systému MATLAB.

1. Popište detailně strukturu "mdl" souboru.
2. Vytvořte MATLABovskou funkci pro generování "mdl" souborů pro simulaci digitálních filtrů dle výsledků návrhu v systému MATLAB.
3. Srovnejte výsledky simulací automaticky generovaných a ručně vytvořených modelů.

Práce bude vytvořena v typografickém systému LaTeX.

Seznam doporučené odborné literatury:

[1] DABNEY, James a Thomas L HARMAN. Mastering SIMULINK. Upper Saddle River: Pearson Prentice Hall, c2004, xix, 376 s. ISBN 0-13-142477-7.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 21. dubna 2016 .....

Rád bych na tomto místě poděkoval mému vedoucímu bakalářské práce, panu Ing. Janu Skapovi, Ph.D., za jeho vstřícný přístup, hodnotné rady a odborné vedení během mé práce.

Abstrakt

Bakalářská práce se zabývá tvorbou modelů v prostředí MATLAB-Simulink s využitím matlabovkského kódu. V úvodní části je rozebrána struktura MDL souboru Simulinku. Je vysvětlena tvorba modelů a dále jsou tyto modely podrobně rozebrány od nejjednodušších variant až po složitější, tvořené digitálními filtry vyšších řádů. Následně jsou tyto poznatky využity k tvorbě funkce pro automatizované generování digitálního IIR filtru přímé nekanonické formy libovolného řádu na základě předchozího výpočtu v prostředí MATLAB. Závěr práce je zaměřen na porovnání a zhodnocení tvorby modelů filtrů ruční a automatizovanou metodou a na porovnání jejich simulací.

Klíčová slova: MATLAB, Simulink, soubor MDL, digitální IIR filtr, model, tvorba modelů

Abstract

Bachelor thesis is focused on creating models for MATLAB-Simulink environment using Matlab code. In the first part, the MDL structure of MDL Simulink file is described followed by the explanation of models creating process. These models are then described in details from the most simple to the more complex ones based on higher-order digital filters. Based on the previous computations in Matlab, this knowledge is used for creating a function for automatic generation of digital IIR filter of direct non-canonical form which can be of any order. The last part of the thesis compares manual and automatic methods for creating filter models and their simulations.

Key Words: MATLAB, Simulink, MDL file, digital IIR filter, model, building models

Obsah

Seznam použitých zkratek a symbolů

ASCII	–	American Standard Code for Information Interchange
FDATOOL	–	Filter Design and Analysis Tool
IIR	–	Infinite Impulse Response
MATLAB	–	MATrix LABoratory
MDL	–	Model File Format
PNG	–	Portable Network Graphics

Seznam obrázků

Úvod

Tato práce se zabývá možnostmi řešení automatizovaného generování modelů v systému Simulink dle výsledků výpočtů v prostředí MATLAB.

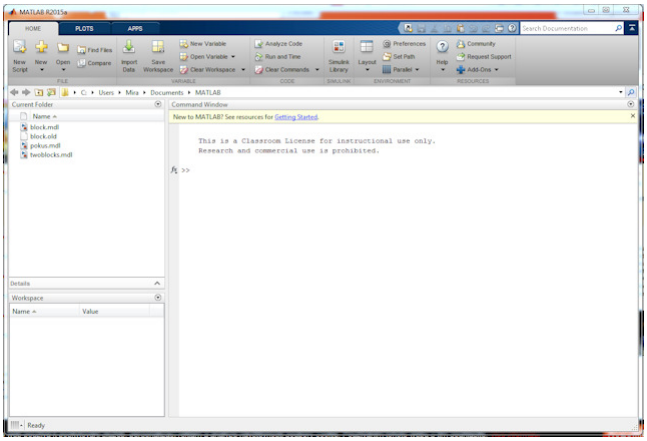
V první kapitole je stručně představena aplikace MATLAB-Simulink.

Druhá kapitola popisuje detailně strukturu souboru MDL. Jsou zde popsány rozdíly mezi modelem obsahujícím jeden blok a modelem obsahujícího dva bloky a jedním propojením.

Třetí a čtvrtá kapitola obsahuje podrobný popis funkcí určených k automatizované tvorbě modelu digitálního filtru buď pomocí příkazů MATLABu sloužících ke tvorbě modelů, nebo přímou tvorbou souboru MDL pomocí příkazu 'fprintf', čehož lze využít ke tvorbě modelů digitálního filtru pomocí open-source programů např. Scilab. Jsou zde obsaženy i ukázky použití těchto funkcí a je zde provedeno porovnání výsledků simulací z obou způsobů tvorby modelu.

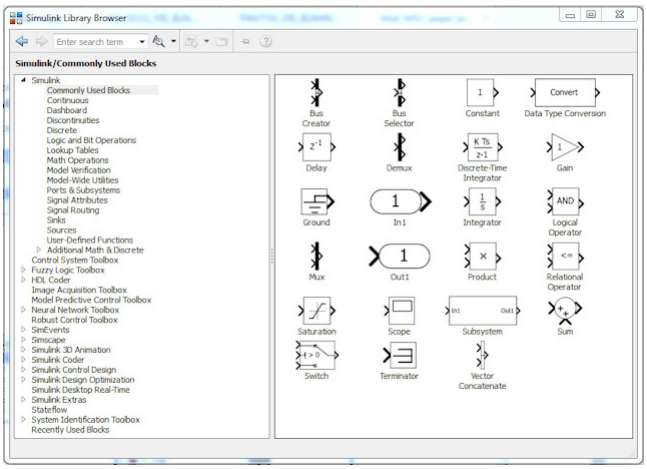
1 Základní popis programu MATLAB - Simulink

MATLAB je výkonný programovací jazyk zahrnující interaktivní prostředí, pomoci kterého můžeme zkoumat, zobrazovat a vytvářet úlohy z různorodých oborů, jako jsou např. výpočetní operace, zpracování a měření signálů. Lze jej využít i k modelovaní, simulacím a analýzám. Použití najde při tvorbě aplikací s vlastním grafickým rozhraním. [1]



Obrázek 1: Ukázka prostředí Matlab

SIMULINK je nástavba MATLABu pro simulaci a modelování systémů formou blokových schémat za použití modelů vytvořených buď pomocí nástrojů grafického prostředí, nebo pomocí Matlab jazyka [1]. Tyto modely jsou vytvořeny z jednotlivých vzájemně propojených bloků, které tvoří součástí systémů. Modely se ukládají do souboru MDL, který je detailně popsán v následující kapitole.



Obrázek 2: Ukázka prostředí Simulink

2 MDL soubor

2.1 Základní informace o MDL souboru a popis jeho struktury

MDL (Model file) soubor je strukturovaný ASCII soubor, který obsahuje popis modelu v prostředí Simulink formou zápisu parametr-hodnota. Tyto zápisy jsou sdruženy do celků umístěných v následující hierarchické struktuře:

```
Model {
   <Model Parameter Name> <Model Parameter Value>
    ...
   BlockDefaults {
      <Block Parameter Name> <Block Parameter Value>
       ...
   }
   AnnotationDefaults {
      <Annotation Parameter Name> <Annotation Parameter Value>
       ...
   }
   System {
      <System Parameter Name> <System Parameter Value>
       ...
      Block {
         <Block Parameter Name> <Block Parameter Value>
           ...
      }
      Line {
         <Line Parameter Name> <Line Parameter Value>
          ...
           Branch {
                <Branch Parameter Name> <Branch Parameter Value>
              ...
         }
      }
      Annotation {
         <Annotation Parameter Name> <Annotation Parameter Value>
          ...
      }
   }
}
```

Výpis 1: Hierarchická struktura MDL souboru [2]

MDL soubor obsahuje tyto části[2]:

- **Model** - definuje parametry modelu
- **BlockDefaults** - obsahuje nastavení jednotlivých bloků **Block** obsažených v modelu
- **AnnotationDefaults** - obsahuje definici poznámek v modelu
- **System** - obsahuje informace o zobrazení modelu a umístění bloků **Block** a jejich propojení **Line** a **Branch** v modelu

2.2 Detailní rozbor modelu obsahujícího jeden blok

Ukažme si, co tvoří MDL soubor, pokud obsahuje pouze jeden blok. V tomto případě byl zvolen generátor sinusového signálu SinWave, viz. obrázek 3. Významy jednotlivých řádků jsou uvedeny v komentáři.

Obrázek 3: Block SinWave

```
Model {
  Name                "block"                    % název modelu
  Version             8.5                        % verze simulinku
  Created             " Fri  Jan 01 16:57:48 2016"  % datum vytvoření
  Creator             "Miroslav"                 % jmeno autora
  LastModifiedBy      "Miroslav"                 % autor poslední úpravy
  LastModifiedDate    " Fri  Jan 01 17:03:24 2016"  % datum poslední změny
  BlockParameterDefaults {
   % v této části  je  nastavení parametru daného bloku, které lze zobrazit  po otevření  bloku
    Block {
      BlockType         Sin                      % typ bloku je  generátor sinus
      SineType          "Time based"             % nastavení SineType
      TimeSource        "Use simulation time"    % nastavení TimeSource
      Amplitude         "1"                      % nastavení amplitudy
      Bias              "0"                      % nastavení Bias
      Frequency         "1"                      % nastavení frekvence
      Phase             "0"                      % nastavení fáze
      Samples           "10"                     % nastavení Samples
      Offset            "0"                      % nastavení offsetu
      SampleTime        "0"                      % nastavení SampleTime
      VectorParams1D    on                       % nastavení VectorParams1D
    }
  }
   % tato část  definuje  umístění pracovní plochy modelu a umístění bloku na pracovní ploše
  System {
    Name                "block"                  % název modelu
    Location            [100,  100, 900, 600]    % umístění okna modelu a rozměr
    Block {
      BlockType         Sin                      % typ bloku
      Name              "Sine Wave"              % název bloku
      SID               "1"                      % ID bloku
      Ports             [0,  1]                  % obsahuje dva porty 0 a 1
      Position          [125,  130, 155, 160]    % pozice a velikost  bloku v  modelu
      ZOrder            1                        % nastavení zobrazovací vrstvy
    }
  }
}
```

Výpis 2: Zobrazení MDL souboru obsahujícího jeden blok SinWave

2.3 Detailní rozbor modelu obsahujícího dva bloky s jedním propojením

V další ukázce máme v modelu umístěny dva bloky SinWave (generátor sinus) a Scope (osciloskop), které jsou spojeny jedním propojením, viz obrázek 4. MDL soubor se v důsledku toho rozrostl o popis druhého bloku a popis vzájemného propojení. Pro větší přehlednost jsou jednotlivé části MDL souboru rozděleny, tak jak následují za sebou, a popsány zvlášť. Jednotlivé bloky mají v kódu odlišeny komentáře barvou. Komentáře týkající se bloku SinWave jsou popsány zeleně, bloku Scope červeně, propojení bloků modře a komentáře kódu určené pro nastavení modelu černě.

Obrázek 4: Blok SinWave a Scope s propojením

Úvodní řádky MDL souboru slouží k uložení parametrů modelu, viz výpis 3, shodně jako u výpisu 2. Je zde pouze uloženo jiné datum vytvoření modelu.

```
Model {
  Name              "twoblocks"              % název modelu
  Version           8.5                      % verze simulinku
  Created           "Sat Jan 02 15:28:42 2016"  % datum vytvoření
  Creator           "Mira"                   % jméno autora
  LastModifiedBy    "Mira"                   % autor poslední úpravy
  LastModifiedDate  "Sat Jan 02 15:32:22 2016"  % datum poslední změny
  .
  .
```

Výpis 3: Část modelu zobrazujícího parametry modelu

Po úvodní části následuje odstavec **BlockParameterDefaults**, který obsahuje nastavení jednotlivých bloků v modelu. Tento odstavec obsahuje dvě části **Block**, každou pro jeden blok. Nejprve je uložena část pro blok Scope viz výpis 4.

```
.
.
  BlockParameterDefaults {
    Block {
      BlockType       Scope                % typ bloku je  osciloskop
      ModelBased      off                  % zapnutí ModelBased
      TickLabels      "OneTimeTick"        % povolení zobrazovacích lišt
      ZoomMode        "on"                 % zoom zapnut na obě souřadnice
      Grid            "on"                 % zobrazení mřížky (do budoucna)
      ShowLegends     off                  % zapnutí legendy průběhu
      TimeRange       "auto"               % nastavení časové základny
      YMin            "-5"                 % nastavení nejnižší úrovně
      YMax            "5"                  % nastavení nejvyšší úrovně
```

```
        SaveToWorkspace off                 % automatické uložení průběhu
        SaveName        "ScopeData"         % jméno pro uložený průběh
        DataFormat      "Array"             % format uložení dat je  do matice
        LimitDataPoints on                  % zapnutí bufferu pro vykreslování
        MaxDataPoints   "5000"              % nastavení bufferu pro vykreslování
        Decimation      "1"                 % ulož každý n-ty sample ("1" ulož každý)
        SampleInput     off                 % nastavení SampleInput
        SampleTime      "-1"                % nastavení SampleTime
        ScrollMode      off                 % nastavení scrolovacího módu
    }
.
.
```

Výpis 4: Část modelu zobrazujícího parametry bloku Scope

Na ni navazuje část s nastavením druhého bloku SinWave, viz výpis 5. Obsah je totožný jako u výpisu 2 s jedním blokem.

```
.
.
Block {
        BlockType       Sin                 % typ bloku je  generátor sinus
        SineType        "Time based"        % nastavení SineType
        TimeSource      "Use simulation time"  % nastavení TimeSource
        Amplitude       "1"                 % nastavení amplitudy
        Bias            "0"                 % nastavení Bias
        Frequency       "1"                 % nastavení frekvence
        Phase           "0"                 % nastavení fáze
        Samples         "10"                % nastavení Samples
        Offset          "0"                 % nastavení offsetu
        SampleTime      "0"                 % nastavení SampleTime
        VectorParams1D  on                  % nastavení VectorParams1D
    }
.
.
```

Výpis 5: Část modelu zobrazujícího parametry bloku SinWave

Další odstavec **System** řeší zobrazení okna modelu a v něm umístěné jednotlivé bloky SinWave, Scope a propojení mezi nimi. Jelikož model obsahuje dva bloky a jedno propojení, má tento odstavec celkem tři části. Dvě části **Block** jsou vyhrazeny pro umístění bloků v modelu, třetí část **Line** slouží k vytvoření propojení.
První řádky odstavce **System** řeší název modelu, umístění okna modelu a jeho velikost, viz výpis 6.

```
.
.
System {
    Name            "twoblocks"         % název modelu
    Location        [100, 100, 900, 600]  % umístění okna modelu a rozměr
.
.
```

Výpis 6: Část modelu řešící umístění okna modelu

Následuje definice pro umístění bloku Scope na pracovní ploše modelu, viz výpis 7.

```
.
.
Block {
    BlockType       Scope                   % typ bloku je osciloskop
    Name            "Scope"                 % název bloku
    SID             "2"                     % ID bloku
    Ports           [1]                     % obsahuje jeden port 1
    Position        [400, 124, 430, 156]    % pozice a velikost bloku v modelu
    ZOrder          2                       % nastavení zobrazovací vrstvy
    Floating        off                     % nastavení parametru Floating
    Location        [188, 365, 512, 604]    % umístění a rozměr okna osciloskopu
    Open            off                     % automaticky otevřít pri startu simulace
    NumInputPorts   "1"                     % počet vstupních portu
  }
.
.
```

Výpis 7: Část modelu řešící umístění bloku Scope

Po něm se nachází umístění bloku SinWave, viz výpis 8

```
.
.
  Block {
    BlockType       Sin                     % typ bloku je generátor sinus
    Name            "Sine Wave"             % název bloku
    SID             "1"                     % ID bloku
    Ports           [0, 1]                  % obsahuje dva porty 0 a 1
    Position        [135, 125, 165, 155]    % pozice a velikost bloku v modelu
    ZOrder          1                       % nastavení zobrazovací vrstvy
  }
.
.
```

Výpis 8: Část modelu řešící umístění bloku SinWave

V poslední části je uloženo propojení bloků. Odstavec **Line** ukládá propojení mezi bloky ve směru ze SrcBlock do DstBlock, v tomto případě z generátoru SinWave do osciloskopu Scope, viz výpis 9.

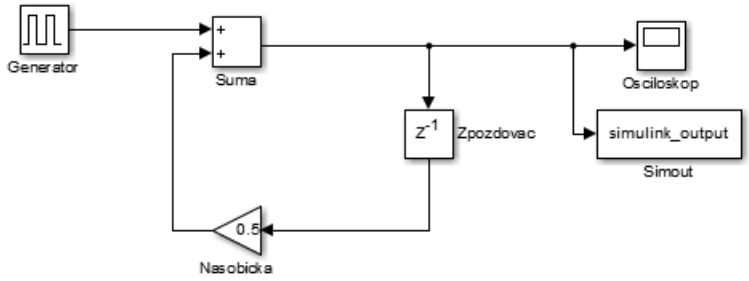
```
.
.
  Line {
    ZOrder          1                       % nastavení zobrazovací vrstvy
    SrcBlock        "Sine Wave"             % zdrojový blok
    SrcPort         1                       % port zdrojového bloku
    DstBlock        "Scope"                 % cílový blok
    DstPort         1                       % port cílového bloku
  }
 }
}
```

Výpis 9: Část modelu řešící umístění propojení

3 Tvorba modelů pomocí jazyka Matlab

3.1 Popis tvorby jednoduchého modelu digitálního filtru

V této kapitole bude vysvětleno vytváření jednoduchého modelu pomocí matlabovského jazyku. Pro ukázku byl vybrán model digitálního filtru 1.řádu, viz obrázek 5. Tento filtr se skládá ze součtového členu, zpožďovače a násobičky. Pro ověření funkce bude ještě připojen na vstup generátor pravoúhlého průběhu a pro kontrolu výstupu osciloskop a blok pro uložení výsledků na Workspace.



Obrázek 5: Blokové zapojení digitálního filtru 1. řádu

Nejprve je potřeba nový model vytvořit. K tomu nám slouží příkaz *new_system*. Musíme mít ale ošetřen případ, pokud budeme spouštet simulaci opakovaně, aby se otevřené okno modelu automaticky zavřelo a byl odstraněn uložený soubor modelu z disku. V opačném případě by totiž došlo při spuštení k chybě a simulace by se nespustila. Tato situace je vyřešena ve výpisu viz 10.

```
% nejprve zkontrolujeme, jestli soubor s modelem již neexistuje, pokud ano, tak jej odstraníme
if exist('filtr','file') == 4
    if bdIsLoaded('filtr')              % kontrola, zda-li je model otevřen
        close_system('filtr',0)         % pokud je otevřen, model se zavře
    end
    delete(['filtr','.mdl'])            % smazání souboru s modelem
end

new_system('filtr')                     % vytvoří nový model
```

Výpis 10: Vytvoření modelu s kontrolou duplicity

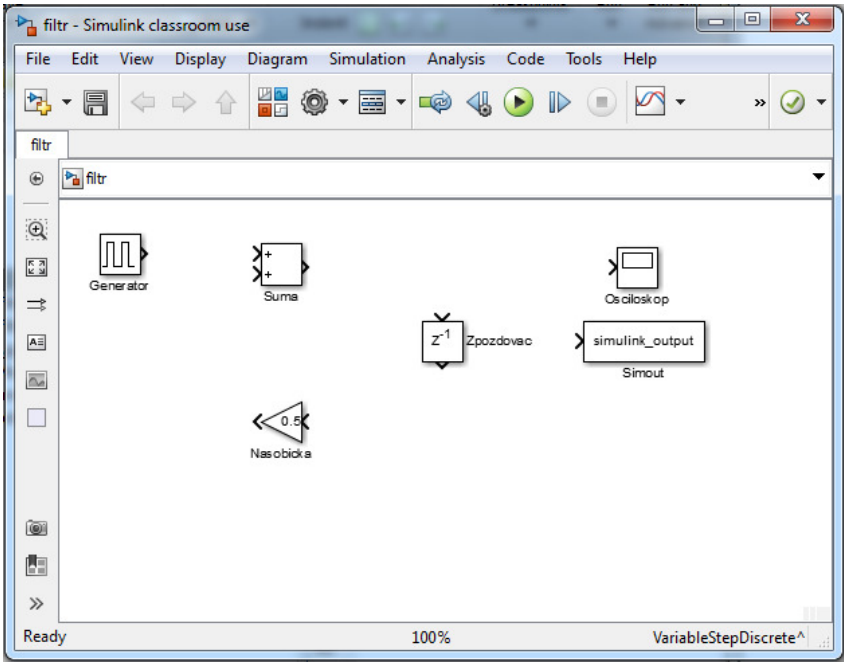
V dalším kroku umístíme potřebné bloky. K tomu slouží příkaz *add_block* viz výpis 11.

```
% pomocné proměnné pro umísťování bloku
x = 30;
y = 30;
w = 30;
h = 30;
offset = 120;
```

```
% vložení bloku Suma na určenou pozici
pozice = [(x+offset) y (x+offset)+w y+h];
add_block('built-in/Sum','filtr/Suma','Position',pozice);

% vložení bloku Delay s nastavením správné orientace
pozice = [(x+offset*2) (y+offset/2) (x+offset*2)+w (y+offset/2)+h];
add_block('built-in/Delay','filtr/Zpozdovac','Position',pozice,'Orientation','down');

% vložení bloku Gain s nastavením správné orientace
pozice = [(x+offset) (y+offset) (x+offset)+w (y+offset)+h];
add_block('built-in/Gain','filtr/Nasobicka','Position',pozice,'Orientation','left');

% vložení bloku Scope
pozice = [(x+25+offset*3) y+5 (x+25+offset*3)+w y+5+h];
add_block('built-in/Scope','filtr/Osciloskop','Position',pozice);

% vložení bloku PulseGenerator
pozice = pozice = [x y-5 x+w y-5+h];
add_block('built-in/Pulsegenerator','filtr/Generator','Position',pozice);

% vložení bloku Simout
pozice = [(x+offset*3) (y+offset/2) (x+offset*3)+w+60 (y+offset/2)+h];
add_block('built-in/Toworkspace','filtr/Simout','Position',pozice);
```

Výpis 11: Vkládání bloků do modelu filtru

Výsledkem je rozmístění bloků na ploše modelu, viz obrázek 6.

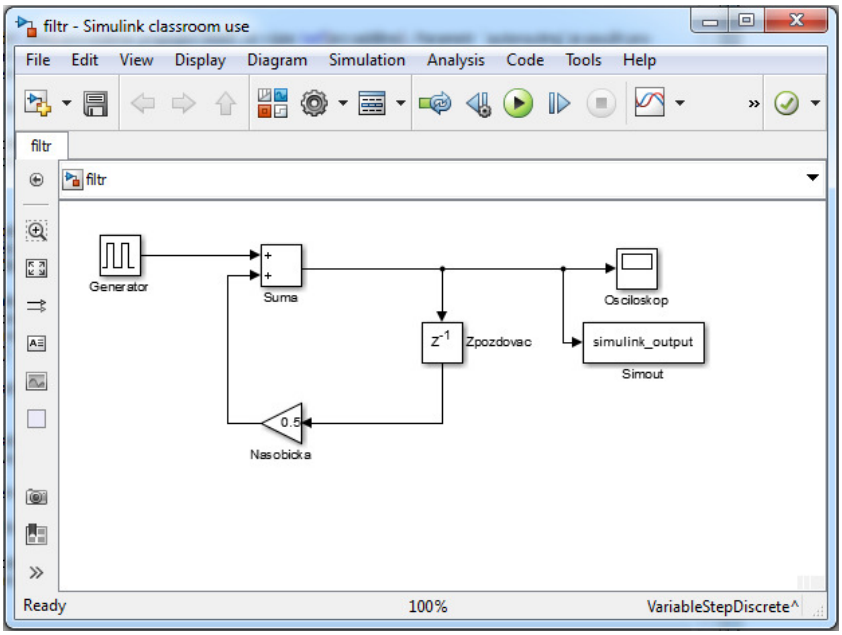


Obrázek 6: Umístění bloků do modelu filtru

Pomocí příkazu *add_line* provedeme propojení bloků viz výpis 12. Parametr 'autorouting' je použit pro vykreslení propojení v pravých úhlech. Výsledek je na obrázku 7.

```
% propojení generátoru do sumy
add_line('filtr','Generator/1','Suma/1','autorouting','on')

% propojení sumy do osciloskopu
add_line('filtr','Suma/1','Osciloskop/1','autorouting','on')

% propojení sumy do zpožďovače
add_line('filtr','Suma/1','Zpozdovac/1','autorouting','on')

% propojení zpožďovače do násobičky
add_line('filtr','Zpozdovac/1','Nasobicka/1','autorouting','on')

% propojení násobičky do sumy
add_line('filtr','Nasobicka/1','Suma/2','autorouting','on')

% propojení sumy do simout
add_line('filtr','Suma/1','Simout/1','autorouting','on')
```

Výpis 12: Vkládání propojení do modelu filtru



Obrázek 7: Umístění propojení do modelu filtru

Pokud požadujeme u bloků změnit nastavení libovolných parametrů, můžeme tak učinit buď přímo při umísťování bloků vložením patřičných parametrů k příkazu *add_block*, nebo můžeme zvolit přehlednější způsob a změnit parametry až po vložení bloku příkazem *set_param*, jak znázorňuje výpis 13.

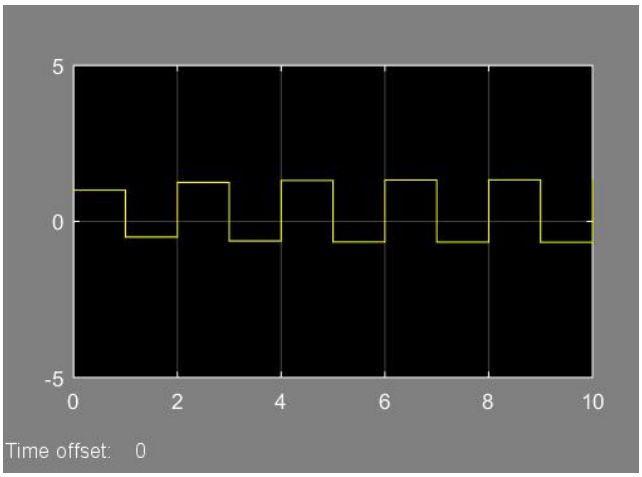
```
% nastavení parametru zpožďovače
set_param(' filtr /Zpozdovac','SampleTime','1','DelayLength','1')
% nastavení parametru násobičky
set_param(' filtr /Nasobicka','Gain','-0.5')
% nastavení rozměru a umístění okna modelu
set_param(gcs,'location',[20  100 600 600])
```

Výpis 13: Nastavení parametrů modelu

Nyní je potřeba model uložit příkazem *save_system* do souboru MDL, otevřít okno modelu pomocí *open_system* a provést spuštění simulace zavoláním *sim*. Příkazem *set_param* nastavíme u osciloskopu otevření okna s průběhem. Vše je zobrazeno ve výpisu 16.

```
save_system(' filtr .mdl')                  % uložení modelu
open_system(' filtr ')                      % otevření okna modelu
sim(' filtr .mdl')                          % spuštění simulace
set_param(' filtr /Osciloskop','Open','on') % otevře okno osciloskopu s vytvořeným průběhem
set(gcf,'Position',[30  60 320 240]);       % nastaví rozměr a umístění okna osciloskopu
```

Výpis 14: Uložení modelu se spuštěním simulace



Obrázek 8: Vyobrazení průběhu z osciloskopu z uloženého obrázku PNG

Grafický průběh osciloskopu si můžeme uložit do souboru PNG, jak je vidět na obrázku 8. K tomu je požit kód ve výpisu 15.

```
% zobraz skryté objekty
set(0,'ShowHiddenHandles','On')
% nastavení velikosti výsledného obrázku
set(gcf,'PaperPositionMode','auto')
% nastavení barev
set(gcf,'InvertHardcopy',' off ')
% uložení do souboru
saveas(gcf,'osciloskop.png')
```

Výpis 15: Funkce pro uložení průběhu osciloskopu do souboru PNG

Pokud požadujeme i uložení obrázku samotného modelu, můžeme tak učinit příkazem $print$. Výsledek je zobrazen na obrázku 5.

```
% uložení zobrazení modelu do souboru png
print (['-s', ' filtr '], '-dpng', ['model_', ' filtr ', '.png']);
```

Výpis 16: Uložení zobrazení modelu do souboru PNG

3.2 Tvorba modelu digitálního IIR filtru 1.řádu přímé nekanonické formy

V této kapitole bude řešen složitější model přímé nekanonické formy digitálního IIR filtru 1.řádu. Budou při tom využity poznatky z předchozí kapitoly a některé části modelu z předcházející kapitoly s malými úpravami využijeme i zde, čímž bude tvorba modelu zjednodušena.

3.2.1 Definice vstupních parametrů filtru

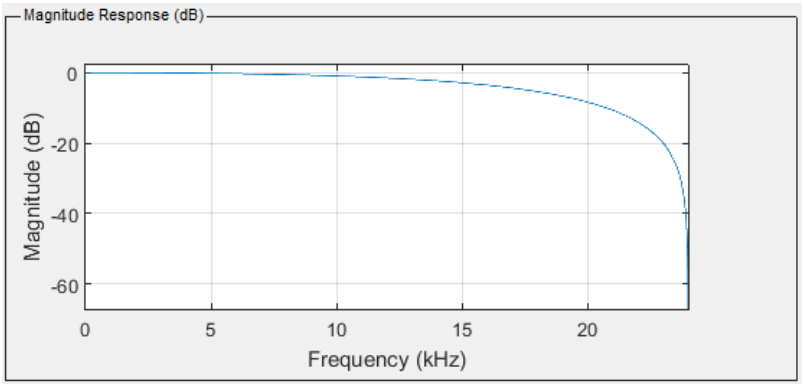
K tomu, abychom mohli vytvořit model zmíněného filtru, musíme nejprve určit, jaké bude mít vstupní parametry. V našem případě zvolíme následující:

f_{vz} = 48000 Hz; f_p = 10000 Hz; f_s = 23000 Hz; A_p = 1dB; A_s = 20dB

K výpočtu potřebných koeficientů násobiček jsem použil grafický nástroj pro návrh digitálních filtrů FDATOOL, který je součástí Matlabu. Jako typ filtru byl zvolen IIR 'Lowpass', metoda výpočtu 'Butterworth', 'match exactly' přepneme na 'Stopband'. Struktura filtru byla nastavena na 'Direct-Form I' a 'Single section'. Tento nástroj mi určil, že se bude jednat o filtr 1.řádu a spočítal potřebné koeficienty násobiček:

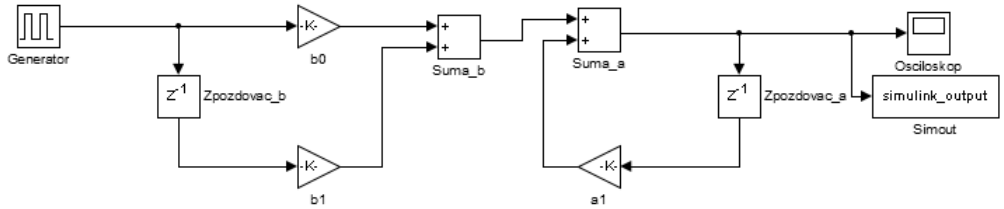
b0 = 0.60527220385197655 a1 = 0.21054440770395313
b1 = 0.60527220385197655

Součásti výpočtu je i graf frekvenčního přenosu filtru, který zobrazuje obrázek 9.



Obrázek 9: Frekvenční přenosová charakteristika IIR filtru 1.řádu

Výsledný model filtru je zobrazen na obrázku 10.



Obrázek 10: Blokové zapojení IIR filtru 1.řádu

3.2.2 Tvorba vlastního filtru

Dříve, než budeme vytvářet nový model, musíme provést, stejně jako v kapitole 3.1, kontrolu, zda-li již nebyl model dříve uložen. K tomu použijeme funkci, které bylo již použito ve výpisu 10, kde pouze upravíme názvy modelu. V tomto případě použijeme pro název modelu označení 'filtr_1r '. Upravený kód je zobrazen ve výpisu 17.

```
% nejprve zkontrolujeme, jestli soubor s modelem již neexistuje,
% pokud ano, tak jej odstraníme
if exist('filtr_1r','file') == 4
    % kontrola, zda-li je model otevřen
    if bdIsLoaded('filtr_1r')
        % pokud je otevřen, model se zavře
        close_system('filtr_1r',0)
    end
    % smazaní souboru s modelem
    delete(['filtr_1r','.mdl']);
end

% vytvoří nový model
new_system('filtr_1r')
```

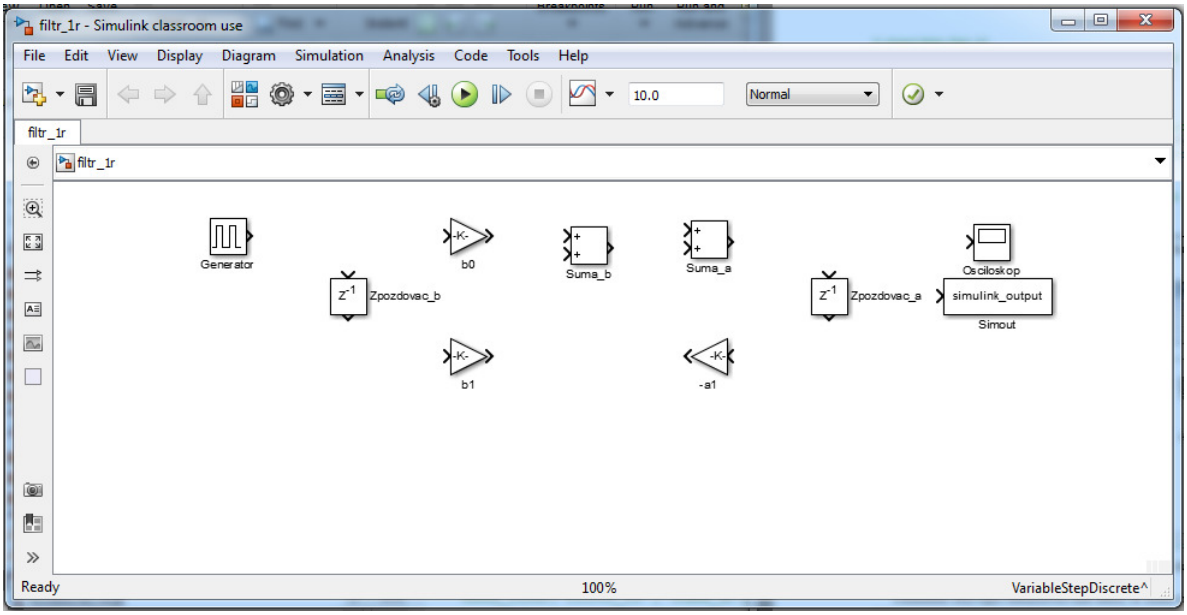
Výpis 17: Vytvoření modelu s kontrolou duplicity

Následně začneme s umísťováním bloků nerekurzivní části filtru. K tomu slouží kód z výpisu 18.

```
% proměnné pro umísťování bloku
x = 30;
y = 30;
w = 30;
h = 30;
offset = 100;

% vložení bloku PulseGenerator
pozice = [x+offset y x+offset+w y+h];
add_block('built-in/Pulsegenerator','filtr_1r/Generator','Position',pozice);

% vložení bloku Gain_b0
% add_block(typ bloku, název v modelu, umístění x y šířka výška)
pozice = [(x+offset*3) (y) (x+offset*3)+w (y)+h];
add_block('built-in/Gain','filtr_1r/b0','Position',pozice);
```

```
% vložení bloku Gain_b1
% add_block(typ bloku, název v modelu, umístění x y šířka výška)
pozice = [(x+offset*3) (y+offset) (x+offset*3)+w (y+offset)+h];
add_block('built-in/Gain','filtr_1r/b1','Position',pozice);

% vložení bloku Delay_b
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*2) (y+offset/2) (x+offset*2)+w (y+offset/2)+h];
add_block('built-in/Delay','filtr_1r/Zpozdovac_b','Position',pozice,'Orientation','down');

% vložení bloku Suma_b na určenou pozici
% add_block(typ bloku, název v modelu, umístění x y šířka výška)
pozice = [(x+offset*4) y+5 (x+offset*4)+w y+5+h];
add_block('built-in/Sum','filtr_1r/Suma_b','Position',pozice);
```

Výpis 18: Vkládání bloků nerekurzivní části filtru



Obrázek 11: Umístění bloků do modelu filtru

Umístíme zbývající rekurzivní část filtru. K tomu můžeme využít část kódu 11 z kapitoly 3.1, pouze upravíme souřadnice bloků a jejich názvy a vypustíme umístění bloku 'PulseGenerator'.

```
% vložení bloku Suma_a na určenou pozici
% add_block(typ bloku, název v modelu, umístění x y šířka výška)
pozice = [(x+offset*5) y (x+offset*5)+w y+h];
add_block('built-in/Sum','filtr_1r/Suma_a','Position',pozice);

% vložení bloku Delay_a
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*6) (y+offset/2) (x+offset*6)+w (y+offset/2)+h];
add_block('built-in/Delay','filtr_1r/Zpozdovac_a','Position',pozice,'Orientation','down');
```

```
% vložení bloku Gain_a1
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*5) (y+offset) (x+offset*5)+w (y+offset)+h];
add_block('built-in/Gain','filtr_1r /-a1','Position',pozice,'Orientation','left');

% vložení bloku Scope
% add_block(typ bloku, název v modelu, umístění x y šířka výška)
pozice = [(x+35+offset*7) y+5 (x+35+offset*7)+w y+5+h];
add_block('built-in/Scope','filtr_1r /Osciloskop','Position',pozice);

% vložení bloku Simout
pozice = [(x+10+offset*7) (y+offset/2) (x+10+offset*7)+w+60 (y+offset/2)+h];
add_block('built-in/Toworkspace','filtr_1r /Simout','Position',pozice);
```

Výpis 19: Vkládání bloků rekurzivní části filtru

Rozmístění bloků máme dokončeno. Pokud bychom kód nyní spustili, vytvoří se nám část modelu, jak je vidět na obrázku 11.

Po umístění bloků zbývá umístit propojení mezi nimi. Nejprve začneme s propojováním v nerekurzivní části, viz výpis 20.

```
% propojení generátoru do násobičky b0
add_line('filtr_1r ','Generator/1','b0/1','autorouting','on')
% propojení násobičky b0 do sumy_b
add_line('filtr_1r ','b0/1','Suma_b/1','autorouting','on')
% propojení generátoru do zpožďovače_b
add_line('filtr_1r ','Generator/1','Zpozdovac_b/1','autorouting','on')
% propojení zpožďovače_b do násobičky b1
add_line('filtr_1r ','Zpozdovac_b/1','b1/1','autorouting','on')
% propojení násobičky b1 do sumy_b
add_line('filtr_1r ','b1/1','Suma_b/2','autorouting','on')

% propojení sumy_b do sumy_a
add_line('filtr_1r ','Suma_b/1','Suma_a/1','autorouting','on')
```

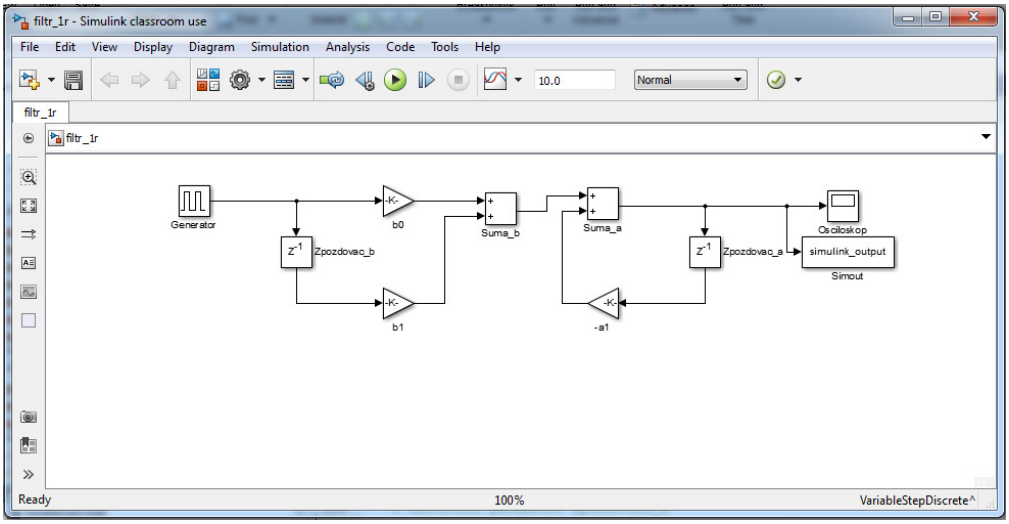
Výpis 20: Vkládání propojení v nerekurzivní části filtru

Posledním krokem tvorby modelu je umístění propojení bloků v rekurzivní části. Opět můžeme využit již vytvořený kód z kapitoly 3.1 z výpisu 12 a pouze aktualizujeme názvy propojovaných bloků, viz výpis 21.

```
% propojení sumy_a do osciloskopu
add_line('filtr_1r ','Suma_a/1','Osciloskop/1','autorouting','on')
% propojení sumy_a do zpožďovače_a
add_line('filtr_1r ','Suma_a/1','Zpozdovac_a/1','autorouting','on')
% propojení zpožďovače_a do násobičky a1
add_line('filtr_1r ','Zpozdovac_a/1','-a1/1','autorouting','on')
% propojení násobičky a1 do sumy_a
add_line('filtr_1r ','-a1/1','Suma_a/2','autorouting','on')
% propojení sumy_a do simout
add_line('filtr_1r ','Suma_a/1','Simout/1','autorouting','on')
```

Výpis 21: Vkládání propojení v rekurzivní části filtru

Model filtru máme tímto hotov. Pokud kód spustíme, vytvoří se nám model na obrázku 12. V dalším kroku přistoupíme k nastavení parametrů jednotlivých bloků, zejména koeficientů násobiček, které máme již spočítány z kapitoly 3.2.1. Nastavení parametrů filtru se věnuje kapitola 3.2.3.



Obrázek 12: Umístění propojení do modelu filtru

3.2.3 Nastavení parametrů modelu

K tomu, aby digitální filtr měl požadovanou funkci, musíme u bloků násobičky a zpožďovače nastavit správné parametry. To provedeme stejným způsobem, jako ve výpisu 12. Část kódu řešící nastavení parametrů bloků je ukázáno ve výpisu 22.

```
% nastavení parametrů zpožďovače_a
set_param('filtr_1r/Zpozdovac_a','SampleTime','1','DelayLength','1')
% nastavení parametrů zpožďovače_b
set_param('filtr_1r/Zpozdovac_b','SampleTime','1','DelayLength','1')
% nastavení parametrů násobičky a1
set_param('filtr_1r/-a1','Gain','-0.21054440770395313')
% nastavení parametrů násobičky b0
set_param('filtr_1r/b0','Gain','0.60527220385197655')
% nastavení parametrů násobičky b1
set_param('filtr_1r/b1','Gain','0.60527220385197655')
% nastavení parametrů osciloskopu
set_param('filtr_1r/Osciloskop','SampleInput','On','SampleTime','125e-6','LimitDataPoints','off')
% nastavení rozměru a umístění okna modelu
set_param(gcs,'location',[20  100 1000 600])
```

Výpis 22: Nastavení parametrů filtru

O uložení modelu, spuštění simulace a otevření okna osciloskopu se stará, stejně jako v dříve uvedeném výpisu 16, následující část kódu 23. Nastavíme pouze jiný název pro model.

```
save_system('filtr_1r.mdl');          % uložení modelu
open_system('filtr_1r')               % otevření okna modelu
sim('filtr_1r.mdl');                  % spuštění simulace

% otevře okno osciloskopu s vytvořeným průběhem
set_param('filtr_1r/Osciloskop','Open','on')
% nastaví rozměr a umístění okna osciloskopu
set(gcf,'Position',[30 60 320 240]);
```

Výpis 23: Uložení modelu se spuštěním simulace

Pokud požadujeme uložení průběhu osciloskopu do souboru PNG, použijeme kód z výpisu 15, pouze na posledním řádku změníme název pro uložený soubor, např. na 'osciloskop_1r.png'. Uložení vzhledu modelu do souboru vykonáme přidáním kódu na posledním řádku. Výsledný kód vidíme ve výpisu 24.

```
% zobraz skryté objekty
set(0,'ShowHiddenHandles','On')
% nastavení velikosti výsledného obrázku
set(gcf,'PaperPositionMode','auto')
% nastavení barev
set(gcf,'InvertHardcopy','off')
% uložení do souboru
saveas(gcf,'osciloskop_1r.png')

% uložení zobrazení modelu do souboru png
print (['-s','filtr_1r'],'-dpng',['model_','filtr_1r','.png']);
```

Výpis 24: Uložení průběhu osciloskopu a zobrazení modelu do souboru PNG

Nyní můžeme v Simulinku matlabovský kód spustit kliknutím na **Run** v liště programu, čímž kód spustíme a dojde k vygenerování modelu filtru viz obrázek 10 a ke spuštění simulace. Zároveň se otevře okno osciloskopu s průběhem, průběh se uloží na disk do PNG souboru a provede se uložení obrázku s vytvořeným modelem do souboru.

3.3 Tvorba modelu digitálního IIR filtru 2.řádu přímé nekanonické formy

K sestavení IIR filtru 2. řádu lze snadno využít již vytvořený kód ke generovaní modelu filtru IIR 1. řádu. K tomu, abychom z něj vytvořili IIR filtr 2.řádu, stačí do modelu přidat na každou stranu filtru (vstupní a rekurzivní) jednu další násobičku se zpožďovačem a přidat do obou sum další vstup, jak bude v této kapitole popsáno.

3.3.1 Definice vstupních parametrů filtru

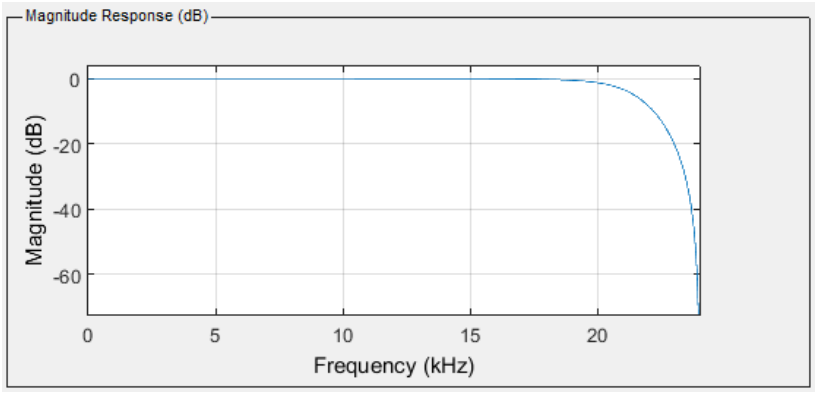
K získání koeficientů násobiček bude opět využit nástroj FDATOOL. Vstupní parametry důležité k výpočtu filtru jsou uvedeny zde:

f_{vz} = 48000 Hz; f_p = 15000 Hz; f_s = 23000 Hz; A_p = 1dB; A_s = 20dB

Jako typ filtru byl opět zvolen IIR 'Lowpass', metoda výpočtu 'Butterworth', 'match exactly' přepneme na 'Stopband'. Struktura filtru byla nastavena na 'Direct-Form I' a 'Single section'. Tentokrát byl nástrojem spočítán filtr 2.řádu s výslednými koeficienty násobiček:

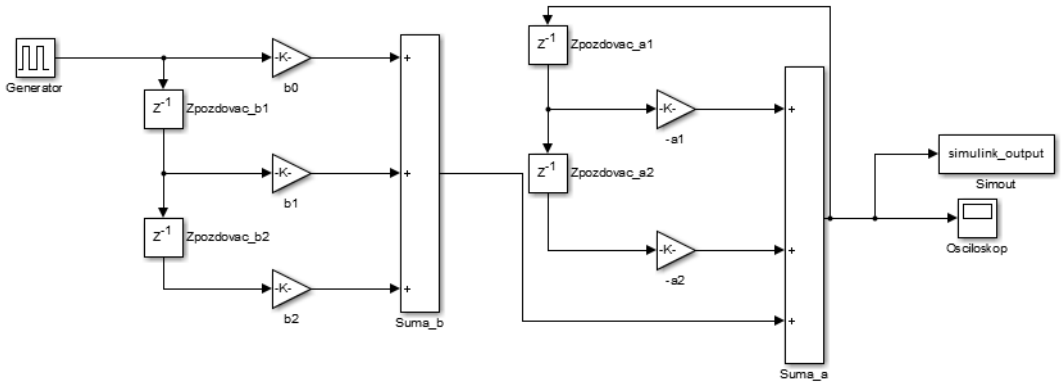
b0 = 0.7489919508967221 a1 = -1.4339539008227333
b1 = 1.4979839017934442 a2 = -0.56201390276415508
b2 = 0.7489919508967221

Graf frekvenčního přenosu filtru je na obrázku 13.



Obrázek 13: Frekvenční přenosová charakteristika IIR filtru 2.řádu

Blokové zapojení filtru znázorňuje obrázek 14.



Obrázek 14: Blokové zapojení IIR filtru 2.řádu

Pro efektivnější práci v tomto případě ještě provedeme vyexportování hodnot násobiček do proměnných Den a Num na Workspace. Tyto proměnné pak využijeme pro automatické nastavení parametrů násobiček na základě výpočtu v FDATOOL.

3.3.2 Tvorba vlastního filtru

Stejně jako u předešlých modelů v kapitolách 3.1 a 3.2.2 je potřeba prověřit, zda již model nebyl vytvořen. Opět využijeme již vytvořený kód, pouze upravíme název modelu. Vzhle-

dem k tomu, že budeme tentokrát využívat proměnné Den a Num, které jsme si vyexportovaly z FDATOOLu, musíme ještě provést kontrolu, zda-li jsou na Workspace opravdu uloženy. Ukázka je obsažena ve výpisu 25.

```
% nejprve zkontrolujeme, jestli soubor s modelem již neexistuje,
% pokud ano, tak jej odstraníme
if exist('filtr_2r','file') == 4
    % kontrola, zda-li je model otevřen
    if bdIsLoaded('filtr_2r')
        % pokud je otevřen, model se zavře
        close_system('filtr_2r',0)
    end
    % smazaní souboru s modelem
    delete(['filtr_2r','.mdl']);
end
% ověříme existenci vstupních proměnných Den a Num
if exist('Den','var') == 0 || exist('Num','var') == 0
   error('Nebyly nalezeny vstupni promenne Den nebo Num. Pouzijte FDATOOL k jejich vygenerovani a
        vyexportujte je na Workspace.')
end
% vytvoříme nový model
new_system('filtr_2r')
```

Výpis 25: Vytvoření modelu s kontrolou duplicity a existence vstupních proměnných

Provedeme rozmístění bloků v nerekurzivní části filtru. K tomu využijeme stejný kód, který byl použit při tvorbě IIR filtru 1.řádu, viz 18 a přidáme do něj navíc řádky řešící umístění další násobičky se zpožďovačem. U řádku starajícího se o umístění sumy přidáme navíc parametr 'Inputs','++', který zajistí přidání 3-vstupé sumy s větším odstupem vstupů mezi sebou pro lepší přehlednost a dále upravíme její rozměr, jak ukazuje výpis 26.

```
% proměnné pro umísťování bloku
x = 30;
y = 30;
w = 30;
h = 30;
offset = 100;

% vložení bloku PulseGenerator
% add_block(typ bloku, název v modelu, umístění x y šířka výška)
pozice = [x+offset y+10 x+offset+w y+10+h];
add_block('built-in/Pulsegenerator','filtr_2r/Generator','Position',pozice);

% vložení bloku Gain_b0
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*3) (y+10) (x+offset*3)+w (y+10)+h];
add_block('built-in/Gain','filtr_2r/b0','Position',pozice);

% vložení bloku Gain_b1
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*3) (y+offset) (x+offset*3)+w (y+offset)+h];
add_block('built-in/Gain','filtr_2r/b1','Position',pozice);
```

```
% vložení bloku Gain_b2
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*3) (y-10+offset*2) (x+offset*3)+w (y-10+offset*2)+h];
add_block('built-in/Gain',' filtr_2r /b2','Position',pozice);

% vložení bloku Delay_b1
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*2) (y+offset/2) (x+offset*2)+w (y+offset/2)+h];
add_block('built-in/Delay',' filtr_2r /Zpozdovac_b1','Position',pozice,'Orientation','down');

% vložení bloku Delay_b2
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*2) (y+offset*1.5) (x+offset*2)+w (y+offset*1.5)+h];
add_block('built-in/Delay',' filtr_2r /Zpozdovac_b2','Position',pozice,'Orientation','down');

% vložení bloku Suma_b na určenou pozici
% add_block(typ bloku, název v modelu, umístění x y šířka výška, vstupy)
pozice = [(x+offset*4) y+5 (x+offset*4)+w y+5+215];
add_block('built-in/Sum',' filtr_2r /Suma_b','Position',pozice,'Inputs','+|+|+');
```

Výpis 26: Vkládání bloků nerekurzivní části filtru

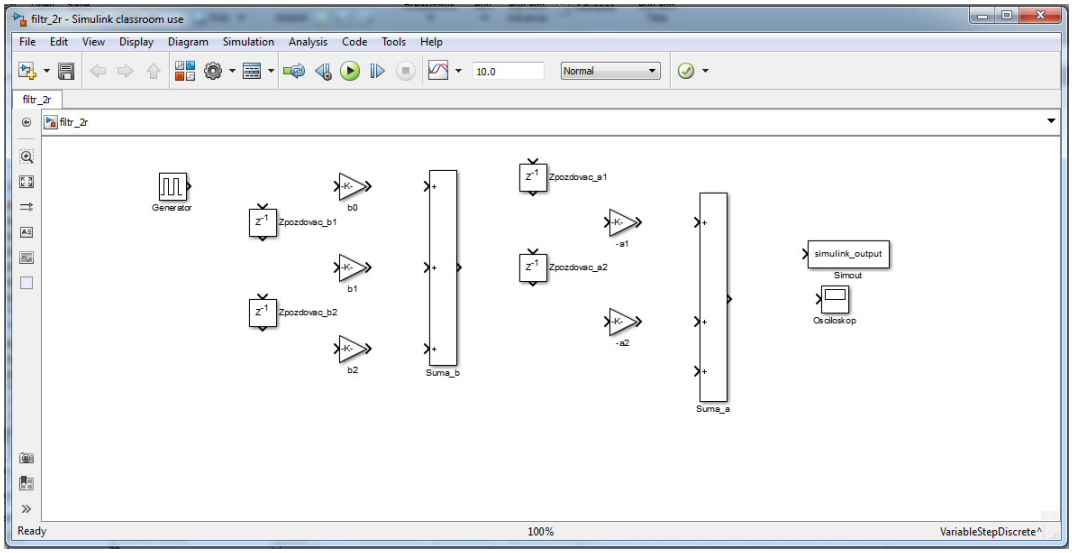
K umístění bloků v rekurzivní části filtru opět využijeme kód z výpisu 19 s přidáním řádků pro umístění další násobičky se zpožďovačem. Opět upravíme počet vstupů u sumy parametrem 'Inputs', tentokrát s hodnotou '+|++', jelikož mezi dvěma spodními vstupy nám stačí menší mezera a upravíme její rozměr. Pro lepší přehlednost výsledného filtru a lepší vykreslení propojovacích spojů, jsem ještě provedl dodatečné úpravy v rozmístění bloků a z toho vyplynula úprava kódu týkající se jejich umístění. Výsledný kód je vypsán ve výpisu 27

```
% vložení bloku Suma_a na určenou pozici
% add_block(typ bloku, název v modelu, umístění x y šířka výška, vstupy)
pozice = [(x+offset*7) y+30 (x+offset*7)+w y+230+h];
add_block('built-in/Sum',' filtr_2r /Suma_a','Position',pozice,'Inputs','+|++');

% vložení bloku Delay_a1
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*5) (y) (x+offset*5)+w y+h];
add_block('built-in/Delay',' filtr_2r /Zpozdovac_a1','Position',pozice,'Orientation','down');

% vložení bloku Delay_a2
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*5) (y+offset) (x+offset*5)+w (y+offset)+h];
add_block('built-in/Delay',' filtr_2r /Zpozdovac_a2','Position',pozice,'Orientation','down');

% vložení bloku Gain_a1
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*6) (y+offset/2) (x+offset*6)+w (y+offset/2)+h];
add_block('built-in/Gain',' filtr_2r /-a1','Position',pozice,'Orientation','right');

% vložení bloku Gain_a2
% add_block(typ bloku, název v modelu, umístění x y šířka výška, orientace)
pozice = [(x+offset*6) (y+10+offset*1.5) (x+offset*6)+w (y+10+offset*1.5)+h];
add_block('built-in/Gain',' filtr_2r /-a2','Position',pozice,'Orientation','right');
```

```
% vložení bloku Scope
% add_block(typ bloku, název v modelu, umístění x y šířka výška)
pozice = [(x+35+offset*8) (y-15+offset*1.5) (x+35+offset*8)+w y+h-15+offset*1.5];
add_block('built-in/Scope',' filtr_2r /Osciloskop','Position',pozice);

% vložení bloku Simout
% add_block(typ bloku, název v modelu, umístění x y šířka výška)
pozice = [(x+20+offset*8) (y-15+offset) (x+20+offset*8)+w+60 (y-15+offset)+h];
add_block('built-in/Toworkspace',' filtr_2r /Simout','Position',pozice);
```

Výpis 27: Vkládání bloků rekurzivní části filtru

Obrázek 15 ukazuje výsledné rozmístění bloků.



Obrázek 15: Umístění bloků do modelu filtru

V dalším kroku doplníme propojení jednotlivých bloků. Zde platí stejně jako u umísťování bloků, že použijeme již jednou vytvořený kód z IIR filtru 1.řádu ve výpisu 20 a dopíšeme pouze řádky pro chybějící propojky přidaných bloků násobičky a zpožďovače.
Jako první umístíme propojení v nerekurzivní části, viz výpis 28.

```
% propojení generátoru do násobičky b0
add_line(' filtr_2r ','Generator/1','b0/1','autorouting','on')
% propojení násobičky b0 do sumy_b
add_line(' filtr_2r ','b0/1','Suma_b/1','autorouting','on')
% propojení generátoru do zpožďovače_b
add_line(' filtr_2r ','Generator/1','Zpozdovac_b1/1','autorouting','on')
% propojení zpožďovače_b do násobičky b1
add_line(' filtr_2r ','Zpozdovac_b1/1','b1/1','autorouting','on')
% propojení násobičky b1 do sumy_b
add_line(' filtr_2r ','b1/1','Suma_b/2','autorouting','on')
% propojení zpožďovače_b1 do zpožďovače_b2
add_line(' filtr_2r ','Zpozdovac_b1/1','Zpozdovac_b2/1','autorouting','on')
```

```
% propojení zpožďovače_b2 do násobičky b1
add_line(' filtr_2r ' , 'Zpozdovac_b2/1','b2/1','autorouting' , 'on')
% propojení násobičky b2 do sumy_b
add_line(' filtr_2r ' , 'b2/1' , 'Suma_b/3','autorouting','on')
% propojení sumy_b do sumy_a
add_line(' filtr_2r ' , 'Suma_b/1','Suma_a/3','autorouting','on')
```

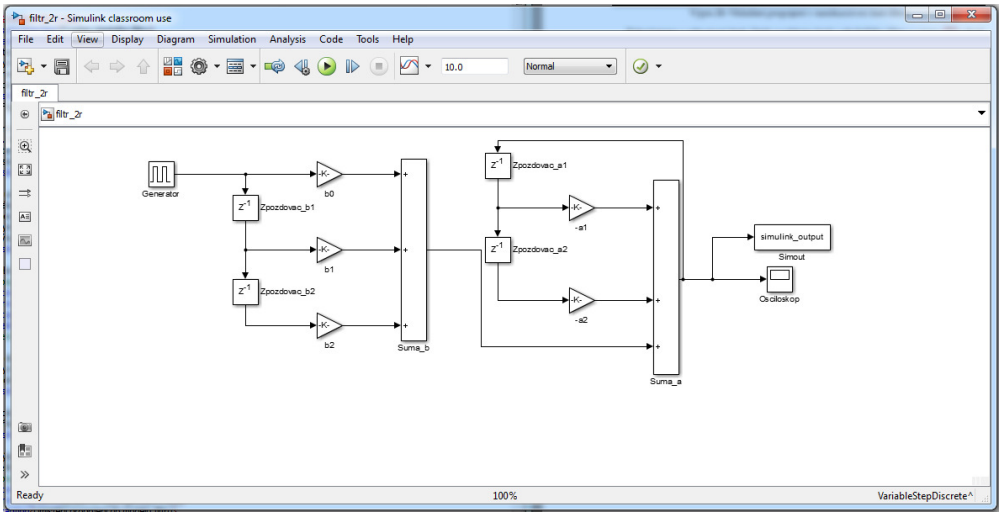
Výpis 28: Vkládání propojení v nerekurzivní části filtru

Pokračujeme v rekuzivní části. Opět využijeme kód z předešlého filtru z výpisu 21 a doplníme propojky pro přidané bloky ve výpisu 29.

```
% propojení sumy_a do zpožďovače_a1
add_line(' filtr_2r ' , 'Suma_a/1','Zpozdovac_a1/1','autorouting','on')
% propojení zpožďovače_a1 do násobičky a1
add_line(' filtr_2r ' , 'Zpozdovac_a1/1','-a1/1','autorouting','on')
% propojení násobičky a1 do sumy_a
add_line(' filtr_2r ' , '-a1/1','Suma_a/1','autorouting','on')
% propojení zpožďovače_a1 do zpožďovače_a2
add_line(' filtr_2r ' , 'Zpozdovac_a1/1','Zpozdovac_a2/1','autorouting','on')
% propojení zpožďovače_a2 do násobičky a1
add_line(' filtr_2r ' , 'Zpozdovac_a2/1','-a2/1','autorouting','on')
% propojení násobičky a2 do sumy_a
add_line(' filtr_2r ' , '-a2/1','Suma_a/2','autorouting','on')
% propojení sumy_a do simout
add_line(' filtr_2r ' , 'Suma_a/1','Simout/1','autorouting' , 'on')
% propojení sumy_a do osciloskopu
add_line(' filtr_2r ' , 'Suma_a/1','Osciloskop/1','autorouting' , 'on')
```

Výpis 29: Vkládání propojení v rekurzivní části filtru

Výsledný filtr je na obrázku 16, který vygenerujeme spuštením kódu. K tomu, aby filtr správně fungoval, provedeme ještě nastavení parametrů jednotlivých bloků (násobičky, zpožďovače) dle výpočtu v kapitole 3.3.1.



Obrázek 16: Umístění propojení do modelu filtru

3.3.3 Nastavení parametrů modelu

K tomu, aby byl digitální filtr nastaven na správné parametry, musíme u bloků násobičky a zpožďovače nastavit správné hodnoty dle výpočtu v kapitole 3.3.1. Opět využijeme původního kódu a vyžijeme ho po malých úpravách i zde. Tentokrát využijeme uložené proměnné Den a Num z Workspace a použijeme je k nastavení násobiček. Upravená verze je obsažena ve výpisu 30.

```
% nastavení parametrů zpožďovače_a1
set_param('filtr_2r/Zpozdovac_a1','SampleTime','125e-6','DelayLength','1')
% nastavení parametrů zpožďovače_a2
set_param('filtr_2r/Zpozdovac_a2','SampleTime','125e-6','DelayLength','1')
% nastavení parametrů zpožďovače_b1
set_param('filtr_2r/Zpozdovac_b1','SampleTime','125e-6','DelayLength','1')
% nastavení parametrů zpožďovače_b2
set_param('filtr_2r/Zpozdovac_b2','SampleTime','125e-6','DelayLength','1')
% nastavení parametrů násobičky a1
set_param('filtr_2r/-a1','Gain',num2str(-Den(2),20))
% nastavení parametrů násobičky a2
set_param('filtr_2r/-a2','Gain',num2str(-Den(3),20))
% nastavení parametrů násobičky b0
set_param('filtr_2r/b0','Gain',num2str(Num(1),20))
% nastavení parametrů násobičky b1
set_param('filtr_2r/b1','Gain',num2str(Num(2),20))
% nastavení parametrů násobičky b2
set_param('filtr_2r/b2','Gain',num2str(Num(3),20))
% nastavení parametrů osciloskopu
set_param('filtr_2r/Osciloskop','SampleInput','On','SampleTime','125e-6','LimitDataPoints','off')
% nastavení rozměru a umístění okna modelu
set_param(gcs,'location',[20  100 1200 700])
```

Výpis 30: Nastavení parametrů filtru

V posledním kroku ještě uložíme model a spustíme simulaci. Stejně jako v předchozím případě pouze upravíme stávající kód z předchozího filtru (výpis 23) tím, že změníme název modelu, viz výpis 31.

```
save_system('filtr_2r.mdl');          % uložení modelu
open_system('filtr_2r')               % otevření okna modelu
sim('filtr_2r.mdl');                  % spuštění simulace

% otevře okno osciloskopu s vytvořeným průběhem
set_param('filtr_2r/Osciloskop','Open','on')
% nastaví rozměr a umístění okna osciloskopu
set(gcf,'Position',[30  60 320 240]);
```

Výpis 31: Nastavení parametrů filtru

K uložení průběhu osciloskopu a zobrazení modelu opět využijeme kód z výpisu 24 a upravíme jména souborů pro aktuální model. Ukázka je ve výpisu 32.

```
% zobraz skryté objekty
set(0,'ShowHiddenHandles','On')
```

```
% nastavení velikosti výsledného obrázku
set(gcf,'PaperPositionMode','auto')
% nastavení barev
set(gcf,'InvertHardcopy','off')
% uložení do souboru
saveas(gcf,'osciloskop_2r.png')
% uložení zobrazení modelu do souboru png
print (['-s', ' filtr_2r '], '-dpng', ['model_', ' filtr_2r ', '.png']);
```

Výpis 32: Uložení průběhu osciloskopu a zobrazení modelu do souboru PNG

3.4 Vzájemné porovnání MDL souborů digitálních IIR filtrů 1. a 2. řádu

V kapitole 2 byl soubor MDL již detailně rozebrán. V této kapitole budou jen stručně popsány rozdíly mezi MDL soubory jednotlivých digitálních IIR filtrů 1. a 2. řádu. Jak bylo již zmíněno v kapitole 3.3.2, filtr 2. řádu vychází z filtru 1.řádu přidáním dalšího páru násobičky a zpožďovače. To je patrné i v MDL souborech těchto filtrů, jak je dále ukázáno.
Následující výpis 33 ukazuje část MDL souboru řešící umístění bloků násobiček IIR filtru 1.řádu. Násobičky jsou celkem tři, dvě jsou v nerekurzivní části (b0, b1), jedna v rekurzivní (a1).

```
System {
  Block {
    BlockType          Gain
    Name               "-a1"
    SID                "8"
    Position           [530, 130, 560, 160]
    ZOrder             8
    BlockMirror        on
    Gain               "-0.21054440770395313"
  }
  Block {
    BlockType          Gain
    Name               "b0"
    SID                "2"
    Position           [330, 30, 360, 60]
    ZOrder             2
    Gain               "0.60527220385197655"
  }
  Block {
    BlockType          Gain
    Name               "b1"
    SID                "3"
    Position           [330, 130, 360, 160]
    ZOrder             3
    Gain               "0.60527220385197655"
  }
  ...
}
```

Výpis 33: Zobrazení části MDL souboru digitálního IIR filtru 1. řádu

IIR filtr 2. řádu obsahuje na každé straně filtru (rekurzivní a nerekurzivní) o jednu násobičku navíc oproti 1. řádu. V nerekurzivní části se jedná o násobičky b0, b1 a b2. Rekurzivní část obsahuje násobičky a1 a a2. Část MDL souboru tohoto filtru je zobrazen ve výpisu 34.

```
System {
    Block {
      BlockType          Gain
      Name          "-a1"
      SID           "11"
      Position          [630, 80, 660, 110]
      ZOrder            11
      Gain          "-1.4339539008227333"
    }
    Block {
      BlockType          Gain
      Name          "-a2"
      SID           "12"
      Position          [630, 190, 660, 220]
      ZOrder            12
      Gain          "-0.56201390276415508"
    }
    Block {
      BlockType          Gain
      Name          "b0"
      SID           "2"
      Position          [330, 40, 360, 70]
      ZOrder            2
      Gain          "0.7489919508967221"
    }
    Block {
      BlockType          Gain
      Name          "b1"
      SID           "3"
      Position          [330, 130, 360, 160]
      ZOrder            3
      Gain          "1.4979839017934442"
    }
    Block {
      BlockType          Gain
      Name          "b2"
      SID           "4"
      Position          [330, 220, 360, 250]
      ZOrder            4
      Gain          "0.7489919508967221"
    }
   ...
}
```

Výpis 34: Zobrazení části MDL souboru digitálního IIR filtru 2. řádu

4 Automatizované generování modelu digitálního filtru MATLABem

Z kapitoly 3.3 vyplývá, že pří tvorbě digitálního filtru vyššího řádů můžeme použít již jednou vytvořený kód a po mírných úpravách parametrů a souřadnic bloků postavit další řád filtru. Za pomoci cyklů, který se bude opakovat vždy s každým řádem filtru, může dojít k postavení celého filtru. Tento poznatek využijeme ve funkci, která filtr vytvoří automaticky na základě vstupních proměnných jako v kapitole 3.3.3, pouze zde doplníme ony cykly tak, abychom mohli vytvořit filtr o libovolném řádu.
Stejně jako v kapitole 3.3 využijeme nástroj FDATOOL, kterým si spočítáme koeficienty násobiček a uložíme je na Workspace do proměnných např. Den a Num, které budou použity jako vstupní proměnné. Název proměnných není tentokrát závazný, jelikož budeme proměnné zadávat jako vstupní parametry funkce pro tvorbu modelu, tudíž může být název jakýkkoliv. Pouze nám musí být známo, jaká proměnná je určena pro nerekurzivní a jaká pro rekurzivní část filtru, jelikož by při případné záměně nedošlo ke správnému vygenerování filtru.
V následujících podkapitolách bude popsána tvorba funkce 'postav_filtr', která bude sloužit pro automatizované generování modelu digitálního filtru s ověřením funkce pomocí generátoru šumu a spektrálního analyzéru. Matlabovský m-soubor musí být pojmenovám stejně, jako název funkce.

4.1 Úvodní část funkce

K tomu, abychom mohli v Matlabu vytvořit funkci, musíme ji v úvodu jako funkci definovat a nastavit u ní vstupní popř. výstupní proměnné. V našem případě bude mít funkce pět vstupních parametrů, z nichž poslední dva nebudou povinné a v případě, že nebudou zadány, nahradí se standartní hodnotou. Funkce má následující vstupní parametry:

- **numIn** - vektor koeficientů nerekurzivní části
- **denIn** - vektor koeficientů rekurzivní části
- **f_vz** - vzorkovací frekvence filtru v Hz
- **simOn** - spuštění simulace filtru (nepovinný) - hodnota 1 pro spuštění
- **'model'** - název vygenerovaného modelu (nepovinný) - zápis jako 'string'

Vlastní funkci definujeme příkazem $function$, kterou je nutno ještě zakončit na posledním řádku funkce pomocí end. Vlastní definici funkce ukazuje výpis 35. Vstupní parametry definujeme pomocí $varargin$, jelikož jejich počet není pevně dán.

```
function postav_filtr (varargin)
```

Výpis 35: Definice funkce

Následuje inicializace proměnných použitých ve funkci, ve výpisu 36.

```
% proměnné pro umísťování bloků
x = 30; y = 30; w = 30; h = 30; offsetX = 100; offsetY = 15;
```

Výpis 36: Inicializace proměnných

Provedeme zpracovaní vstupních argumentů a předáme je do proměnných. Pokud nebudou zadány argumenty **simOn** a **model**, nahradíme je hodnotami **simOn** = 0 a **model** = 'autoCreatedFilter'. Způsob zpracování argumentů zobrazuje výpis 39

```
% přiřazení argumentu proměnným dle jejich zadaného počtu
switch nargin
    % zadány tři argumenty
    case 3
        numIn = varargin{1};
        denIn = varargin{2};
        f_vz = varargin{3};
        simOn = 0;
        model = 'autoCreatedFilter';
    % zadány čtyři argumenty
    case 4
        numIn = varargin{1};
        denIn = varargin{2};
        f_vz = varargin{3};
        simOn = varargin{4};
        model = 'autoCreatedFilter';
    % zadáno pět argumentů
    case 5
        numIn = varargin{1};
        denIn = varargin{2};
        f_vz = varargin{3};
        simOn = varargin{4};
        model = varargin{5};
    % zadán jiný počet argumentu
    otherwise
        error('Spatny pocet vstupnich argumentu!')
end
```

Výpis 37: Předání vstupních argumentů do proměnných

K tomu, aby funkce pracovala korektně, musíme zajistit validitu vstupních dat. Například ověříme, zda-li jsou argumenty **numIn** a **denIn** vektory a zda obsahují číselné hodnoty. Kompletní ověření validace ukazuje výpis 38.

```
% test na korektnost vstupních argumentů numIn a denIn
if ~isvector(numIn); error('Vstupni argument numIn neni vektor!'); end;
if ~isnumeric(numIn); error('Vstupni argument numIn neni ciselny vektor!'); end;
if ~isvector(denIn); error('Vstupni argument denIn neni vektor!'); end;
if ~isnumeric(denIn); error('Vstupni argument denIn neni ciselny vektor!'); end;
% ověřeni korektnosti vstupního argumentu f_vz
if ~isnumeric(f_vz); error('Vstupni argument f_vz neni ciselny!'); end;
% oveření korektnosti vstupního argumentu simOn
if ~(simOn == 1 || simOn == 0); error('Spatna hodnota argumentu simOn'); end
```

Výpis 38: Test validity vstupních dat

Dále doplníme zbývající argumenty do příslušných proměnných, výpis 39.

```
% název souboru pro uložení modelu
 file  = [model '.mdl'];
% perioda vzorkování
sampleTime = 1 / f_vz;
```

Výpis 39: Zpracování vstupních argumentů

V další části ošetříme, zda-li není model filtru již otevřen, popřípadě zda-li neexistuje uložený model se stejným názvem, jak ukazuje výpis 40. Zároveň vytvoříme model filtru.

```
% kontrola, zda-li neni model otevřen
 if bdIsLoaded(model)
     % pokud je otevřen, model se zavře
     close_system(model,0)
end

% zkontrolujeme, jestli soubor s modelem již neexistuje,
 if exist( file ,' file ') == 4
     % pokud ano, tak jej odstraníme
     delete( file );
end

% vytvoří nový model
new_system(model);
```

Výpis 40: Vytvoření modelu s kontrolou duplicity

4.2 Automatizované generovaní nerekurzivní části filtru

V této podkapitole je rozebrána čast funkce řešící tvorbu nerekurzivní (vstupní) části filtru, která navazuje na úvodní část z kapitoly 4.1.
Nejprve, než začne samotná realizace této části, musíme inicializovat proměnnou zodpovědnou za počet cyklů provádějících vlastní tvorbu nerekurzivní (vstupní) části filtru, podle řádu filtru. K tomu slouží proměnná **orderB**, do níž se uloží velikost vektoru vstupní proměnné **numIn**. V případě filtru 1. řádů se na konci vektoru vyskytuje nežádoucí nula, kterou zanedbáme. Realizace je ve výpisu 41.

```
% zjištění řádu filtru nerekurzivní části
orderB = length(numIn);

% eliminace nuly na konci vektoru
 if numIn(orderB) == 0
     % pokud se nula vyskytuje, provede se dekrementace řádu filtru
     orderB = orderB - 1;
end
```

Výpis 41: Zjištění řádu filtru nerekurzivní části

Nyní můžeme začít s tvorbou vlastního modelu. Jako první umístíme sumu. V první řadě musíme nastavit počet vstupů podle řádu filtru a následně sumu umístíme, viz výpis 42.

```
% generovaní vstupu 'sumy b' podle řádu filtru
bSumIn = '+|+';
if orderB > 2
    for i = 3 :orderB
        bSumIn = strcat(bSumIn, '|+');
    end
end

% umístění 'sumy b'
posSum = [(x+offsetX*4) y+5 (x+offsetX*4)+w y+(offsetX*orderB)];
add_block('built-in/Sum',[model '/Suma b'],'Position',posSum,'Inputs',bSumIn);
```

Výpis 42: Umístění sumy s nastavením počtu vstupů

Za účelem zjištění frekvenční přenosové charakteristiky umístíme generátor šumu a nastavíme vzorkovací periodu, viz výpis 45.

```
posGen = [x+offsetX y+15 x+offsetX+w y+15+h];
add_block('built-in/Reference',[model '/Generator sumu'],'SourceBlock',...
    'simulink/Sources/Band-Limited White Noise','Position',posGen);
set_param([model '/Generator sumu'],'Ts',num2str(sampleTime,20))
```

Výpis 43: Umístění generátoru šumu s nastavením parametrů

V další části funkce provedeme umístění zpožďovačů a násobiček. K tomu využijeme již zmiňovaný cyklus, který se provádí tolikrát, dokud nedojde k vytvoření všech řádů. Při tom se zároveň ukládájí koeficienty do jednotlivých násobiček, vzorkovací perioda do zpožďovačů a provede se propojení násobiček se zpožďovači a sumou. Zároveň se propojí bloky vyššího řádu s nižším. Výpis kódu obsahuje výpis 44.

```
% počet opakování je roven řádu filtru + 1 (1. řád filtru má dva průchody cyklem)
for i = 1 :orderB, label = int2str(i-1);
    % umístění násobiček s nastavením parametrů
    posGain = [(x+offsetX*3) (y+offsetY) (x+offsetX*3)+w (y+offsetY)+h];
    add_block('built-in/Gain',[model '/b' label], 'Position',posGain);
    set_param([model '/b' label], 'Gain',num2str(numIn(i),20))
    % propojení násobiček se sumou
    add_line(model,['b' label '/1'],['Suma b/' int2str(i)], 'autorouting','on');

    if i > 1    % podmínka řeší umístění zpožďovače až po druhém cyklu
        label_p = int2str(i-2);
        % umístění zpožďovačů s nastavením parametrů
        posDelay = [(x+offsetX*2) ((y+offsetY)-55) (x+offsetX*2)+w ((y+offsetY)-55)+h];
        add_block('built-in/Delay',[model '/Zpozdovac b' label], 'Position',posDelay,...
            'Orientation','down');
        set_param([model '/Zpozdovac b' label],'SampleTime',num2str(sampleTime,20),...
            'DelayLength','1','ShowName','off')
        % propojení zpožďovačů s násobičkami
        add_line(model,['Zpozdovac b' label '/1'],['b' label '/1'], 'autorouting','on')
```

```
        if  i > 2    % pokud je řád  filtru  větší  než 2, je  třeba propojit  zpožďovač s nižším řádem
            add_line(model,['Zpozdovac b' label_p '/1' ],[ 'Zpozdovac b' label '/1' ],...
                'autorouting', 'on')
        end
    end
    % na konci cyklu nastavíme offsetY k tvorbě další  řady bloků
    offsetY = offsetY + 110;
end
```

Výpis 44: Umístění násobiček se zpožď'ovači a propojkami mezi nimi

Na závěr propojíme generátor šumu se vstupem filtru a ještě provedeme korekci zobrazení sumy, aby propojení násobiček s jednotlivými vstupy sumy byly zobrazeny korektně. Kód je zobrazem ve výpisu 45.

```
% korekce pozice 'sumy b' podle řádu  filtru
posSum = [(x+offsetX*4) y+5 (x+offsetX*4)+w posGain(4)+10];
set_param([model '/Suma b'],'Position',posSum)
% propojení generátoru šumu s první násobičkou
add_line(model,'Generator sumu/1','b0/1','autorouting', 'on');
% propojení generátoru šumu s prvním zpožďovačem
add_line(model,'Generator sumu/1','Zpozdovac b1/1','autorouting','on');
```

Výpis 45: Propojení generátoru šumu se vstupem filtru a korekce zobrazení sumy

4.3 Automatizované generovaní rekurzivní části filtru

Následující podkapitola popíše tvorbu rekurzivní části filtru a navazuje na část nerekurzivní, která byla popsána v podkapitole 4.2. U této části navíc přibude v kódu podmínka ošetřující stav, že pokud bude vstupní vektor obsahovat pouze jednu položku s hodnotou '1', nebude se tato část filtru vůbec generovat a bude nahrazena propojením.
Stejně jako v úvodu podkapitoly 4.2 zinicializujeme proměnnou pro řád rekurzivní části filtru **orderA** a nastavíme pomocnou proměnnou **offsetY** na výchozí hodnotu, jak je ve výpisu 46 ukázáno.

```
offsetY = 15;
% zjištění řádu  filtru  rekurzivní části
orderA = length(denIn);
```

Výpis 46: Zjištění řádu filtru rekurzivní část

Jak již bylo zmíněno v úvodu této podkapitoly, je potřeba ošetřit možost, že rekurzivní část filtru nebude potřeba generovat, pokud bude vstupní vektor obsahovat pouze jeden prvek s hodnotou '1'. Toho docílíme podmínkou ve výpisu 47, která bude platit pro další části kódu a je zakončena až ve výpisu 52. Pro přehlednost je tato část rozdělena na více části.

```
if  ~((orderA == 1) && (denIn(1) == 1))
```

Výpis 47: Podmínka pro případ umístění propojení místo filtru

Dále musíme, stejně v jako nerekurzivní části, elimimovat nulu na konci vektoru. To vyřešíme podobně jako ve výpisu 41, pouze změníme název proměnné, viz výpis 48.

```
% eliminace nuly na konci vektoru
    if denIn(orderA) == 0
        % pokud se nula vyskytuje, provede se dekrementace řádu filtru
        orderA = orderA - 1;
    end
```

Výpis 48: Korekce řádu rekurzivní části filtru

Provedeme umístění sumy s nastavením správného počtu vstupů, viz výpis 49. Kvůli lepšímu zobrazení výsledného filtru je spodní vstup nastaven na menší vzdálenost.

```
if orderA > 2
        aSumIn='+|+';
        for i = 3 :orderA
            if i < orderA
            aSumIn = strcat(aSumIn, '|+');
            else
                aSumIn = strcat(aSumIn, '+');
            end
        end
else aSumIn = '++';
end

% umístění 'sumy a'
posSum = [(x+offsetX*7) y+50 (x+offsetX*7)+w y+(offsetX*orderB)];
add_block('built-in/Sum',[model '/Suma a'],'Position',posSum,'Inputs',aSumIn);
```

Výpis 49: Umístění sumy s nastavením počtu vstupů

Následuje umístění násobiček a zpožďovačů, k čemuž opět využijeme generování bloků v cyklu, jak je zobrazeno ve výpisu 50. Zároveň se provede i propojení násobiček se sumou a zpožďovači a nastaví se koeficienty násobiček a parametry zpožďovačů. Propojíme také bloky vyššího řádu s nižším.

```
% počet opakování je roven řádu filtru
for i = 2 :orderA, label = int2str(i-1); label_p = int2str(i-2);
    % umístění násobiček s nastavením parametrů
    posGain = [(x+offsetX*6) (y+offsetY+50) (x+offsetX*6)+w (y+offsetY+50)+h];
    add_block('built-in/Gain',[model '/-a' label], 'Position',posGain);
    set_param([model '/-a' label], 'Gain',num2str(-denIn(i),20))
    % umístění zpožďovačů s nastavením parametrů
    posDelay = [(x+offsetX*5) (y+offsetY) (x+offsetX*5)+w y+offsetY+h];
    add_block('built-in/Delay',[model '/Zpozdovac a' label], 'Position',posDelay,...
        'Orientation', 'down');
    set_param([model '/Zpozdovac a' label],'SampleTime',num2str(sampleTime,20),...
        'DelayLength','1', 'ShowName','off')
    % propojení zpožďovačů s násobičkami
    add_line(model,['Zpozdovac a' label '/1' ],[ '-a' label '/1' ], 'autorouting', 'on')
    % propojení násobiček se sumou
    add_line(model,['-a' label '/1' ],[ 'Suma a/' int2str(i-1)],'autorouting', 'on');
```

```
    % pokud je řád filtru větší než 2, je třeba propojit zpožďovač s nižším řádem
    if i > 2,
        add_line(model,['Zpozdovac a' label_p '/1' ],[ 'Zpozdovac a' label '/1' ],...
            'autorouting', 'on')
    end
    % na konci cyklu nastavíme offsetY k tvorbě další řady bloků
    offsetY = offsetY + 110;
end
```

Výpis 50: Umístění násobiček se zpožďovači a propojkami mezi nimi

Tímto máme rekurzivní část filtru vygenerovánu a zbývá ji propojit s částí nerekurzivní. Propojíme první zpožďovač s výstupem sumy a umístíme spektrální analyzér s propojením na výstup sumy, viz výpis 51.

```
% propojení 'sumy b' se 'sumou a'
add_line(model,'Suma b/1',['Suma a/' int2str(i)], 'autorouting', 'on')
% korekce pozice 'sumy a' podle řádu filtru
posSum = [(x+offsetX*7) y+50 (x+offsetX*7)+w posGain(4)+70];
set_param([model '/Suma a'],'Position',posSum)
% propojení prvního zpožďovače s výstupem 'sumy a'
add_line(model,'Suma a/1','Zpozdovac a1/1','autorouting','on')
% umístění spektrálního analyzéru
posAna = [(x+35+offsetX*8) (y+15) (x+35+offsetX*8)+w y+h+15];
add_block('built-in/Reference',[model '/Spektralni Analyzer'], 'SourceBlock',...
    'simulink_extras/Additional Sinks/Spectrum Analyzer','Position',posAna);
% propojení 'sumy a' se spektrálním analyzérem
add_line(model,'Suma a/1','Spektralni Analyzer/2', 'autorouting', 'on')
```

Výpis 51: Umístění spektrálního analyzéru s propojením

Jestliže se nebude generovat rekurzivní část viz podmínka ve výpisu 47, umístí se spektrální analyzér blíže k nerekurzivní části a propojí se s výstupem její sumy.

```
else
        posAna = [(x+35+offsetX*5) (y+15) (x+35+offsetX*5)+w y+h+15];
        add_block('built-in/Reference',[model '/Spektralni Analyzer'], 'SourceBlock',...
            'simulink_extras/Additional Sinks/Spectrum Analyzer','Position',posAna);
        add_line(model,'Suma b/1','Spektralni Analyzer/2', 'autorouting', 'on')
end             % ukončení podmínky pro tvorbu rekurzivní části
```

Výpis 52: Druhý způsob umístění spektrálního analyzéru s propojením

V poslední části ještě propojíme druhý vstup spektrálního analyzéru s generátorem šumu, nastavíme jeho vzorkovací periodu a další parametry. To provedeme pomocí kódu ve výpisu 53.

```
% nastavení parametrů spektr. analyzéru
set_param([model '/Spektralni Analyzer'], 'sampleT',num2str(sampleTime,20),...
    'npts', '1024',' fftpts ', '1024','HowOften','256')
% propojení druhého vstupu spektr. analyzéru s generátorem šumu
add_line(model,'Generator sumu/1','Spektralni Analyzer/1', 'autorouting', 'on')
```

Výpis 53: Nastavení parametru spektrálního analyzéru a propojení s generátorem šumu

4.4 Závěrečná část funkce

V závěrečné fázi funkce ještě nastavíme velikost okna modelu a uložíme právě vygenerovaný filtr do souboru, který otevřeme v okně modelu a spustíme simulaci, pokud to bylo vstupním argumentem **simOn** vyžadováno. Během simulace se nám otevře okno spektrálního analyzéru, na kterém můžeme sledovat výslednou přenosovou charakteristiku digitálního filtru. Zároveň provedeme uložení obrázku modelu do souboru PNG. Kód je zobrazen ve výpisu 54.

```
set_param(gcs,'location',[20  20 1200 900]);      % nastavení rozměru a umístění okna modelu
set_param(model,'Solver','ode15s','StopTime','0.2');% korekce času simulace modelu
save_system(model,file);                          % uložení modelu
open_system(model);                               % otevření okna modelu
 if  simOn == 1; sim(model); end                  % spuštění simulace, bylo−li zadano argumentem
% uložení zobrazeni modelu do souboru png
 print ([ '−s',  model], '−dpng', ['model_', model, '.png']) ;
end                                               % zakončení funkce
```

Výpis 54: Závěrečná část funkce

4.5 Ukázka použití funkce

K tomu, abychom mohli vygenerovat digitální filtr pomocí právě vytvořené funkce 'postav_filtr', musíme nejprve pomocí FDATOOLu, který je součásti Matlabu, vypočítat oba vektory, které potřebujeme ke spuštění funkce a uložit si je na Workspace. K výpočtu použijeme tyto vstupní parametry:

f_{vz} = 48000 Hz; f_p = 13000 Hz; f_s = 15000 Hz; A_p = 1dB; A_s = 20dB

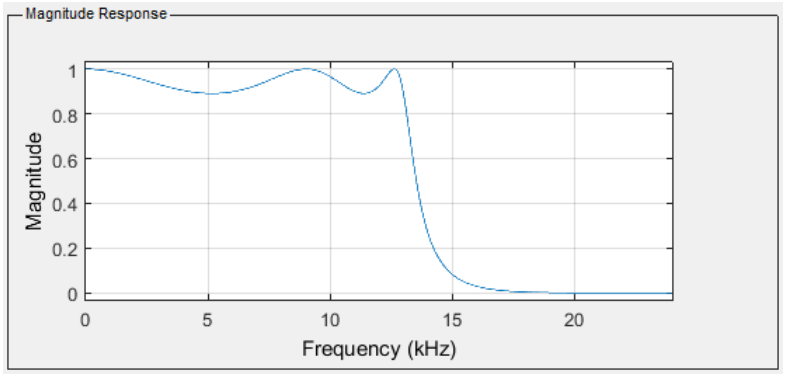
Jako typ filtru je zvolen IIR 'Lowpass', metoda výpočtu 'Chebyshev Type I', 'match exactly' přepneme na 'Stopband'. Strukturu filtru nastavíme na 'Direct-Form I' a 'Single section'. Kompletní nastavení aplikace FDATOOL je na obrázku 19. Následně je spočítáno, že se jedná o filtr 5. řádu s těmito koeficienty:

b0 = 0.034182615960929759	a1 = -0.69691815197801921
b1 = 0.17091307980464879	a2 = 1.3261302895690603
b2 = 0.34182615960929763	a3 = -0.8598530954583723
b3 = 0.34182615960929763	a4 = 0.53058044883917554
b4 = 0.17091307980464879	a5 = -0.20609578022209246
b5 = 0.034182615960929759	

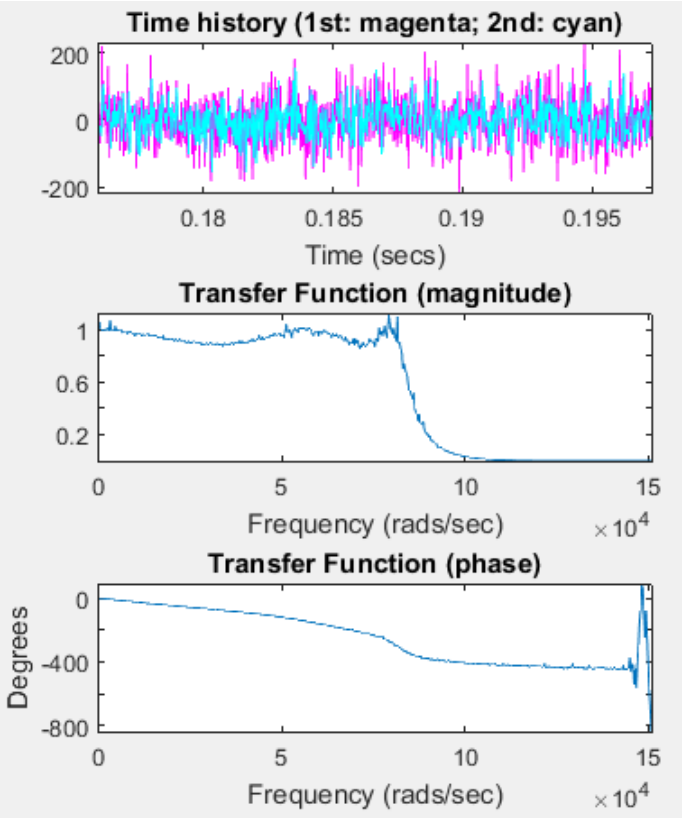
Oba vektory s koeficienty vyexportujeme na Workspace. Vektor pro nerekurzivní část je pojmenován jako 'Num' a pro rekurzivní jako 'Den'. Vzorkovací frekvenci máme 48000Hz. Nyní můžeme začít s generováním filtru. Do 'Command window' tedy zadáme následující příkaz:

postav_filtr(Num,Den,48000,1)

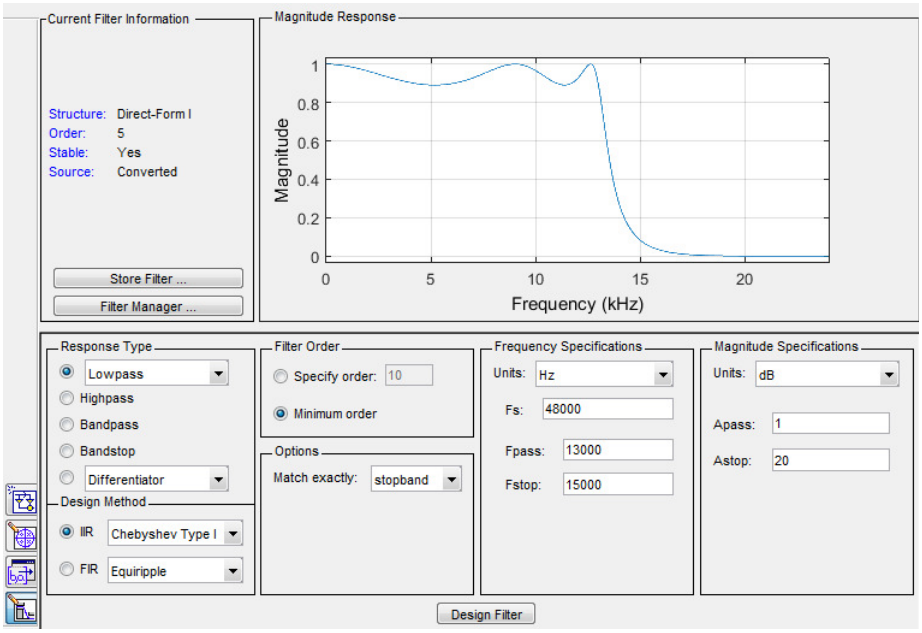
Tímto se nám automaticky vygeneruje digitální IIR filtr 5. řádu viz obrázek 20 a zobrazí se jeho přenosová charakteristika v otevřeném okně spektrálního analyzéru viz obrázek 22. Pro porovnání je na obrázku 17 zobrazena frekvenční charakteristika vygenerovaná FDATOOLem. Je patrné, že obě charakteristiky jsou totožné.



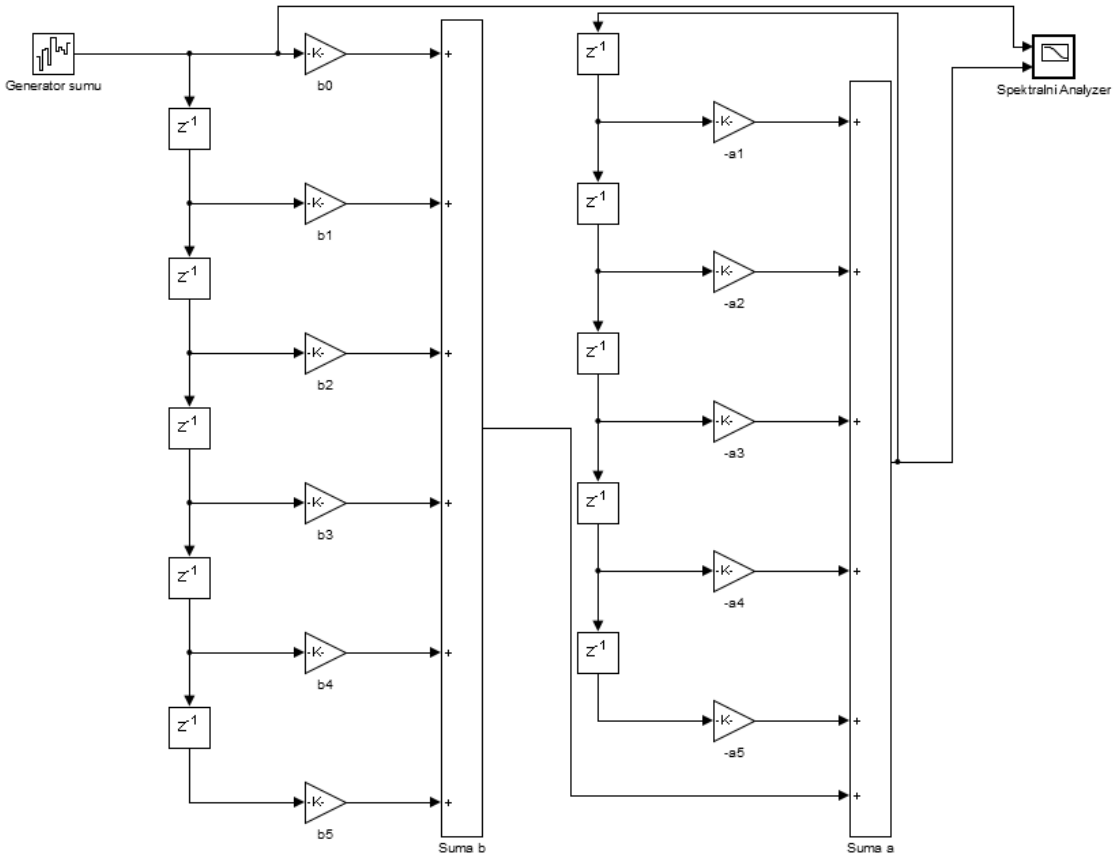
Obrázek 17: Frekvenční přenosové charakteristika vygenerované FDATOOLem



Obrázek 18: Frekvenční přenosové charakteristika zobrazená spektrálním analyzérem



Obrázek 19: Nastavení parametrů filtru v nástroji FDATOOL



Obrázek 20: Vygenerovaný digitální IIR filtr 5. řádu

5 Automatizované generování modelu digitálního filtru přímou tvorbou souboru MDL

Další metodou, jak lze vytvořit model digitálního filtru, je přímá tvorba souboru MDL. To lze využít pro případ, pokud budeme potřebovat vytvořit model filtru v jiné aplikaci než MATLABu a pak si přes MATLAB jen spustit simulaci. Pro generování bude použit stejný algoritmus, jako v případě generování modelu filtru v kapitole 4, pouze budou nahrazeny příkazy add_block, add_line a set_param vkládáním jednotlivých vygenerovaných částí přímo do MDL souboru. Použití a vstupní argumenty funkce buou totožné.
Stejně jako v kapitole 4 využijeme nástroj FDATOOL, který použijeme k výpočtu koeficientů násobiček a ty vzápětí uložíme na Workspace do proměnných např. Den a Num, které budou použity jako vstupní proměnné.
V následujících podkapitolách bude popsána tvorba funkce 'postav_mdl', která bude sloužit pro automatizované generování modelu digitálního filtru přímou tvorbou MDL souboru s ověřením funkce pomocí generátoru šumu a spektrálního analyzéru. Matlabovský m-soubor musí být pojmenovám stejně, jako název funkce.

5.1 Úvodní část funkce

Funkce má opět 5 vstupních parametrů, z nichž poslední dva nebudou povinné a v případě, že nebudou zadány, nahradí se standartní hodnotou. Funkce má následující vstupní parametry:

- **numIn** - vektor koeficientů nerekurzivní části
- **denIn** - vektor koeficientů rekurzivní části
- **f_vz** - vzorkovací frekvence filtru v Hz
- **simOn** - spuštění simulace filtru (nepovinný) - hodnota 1 pro spuštění
- **'model'** - název vygenerovaného modelu (nepovinný) - zápis jako 'string'

Funkci definujeme podobně, jako ve výpisu35, pouze funkci pojmenujeme jako $postav\_mdl$. Vstupní parametry definujeme opět pomocí $varargin$, jelikož jejich počet není pevně dán. Definice funkce ukazuje výpis 55.

```
function postav_mdl(varargin)
```

Výpis 55: Definice funkce

V další části se provede inicializace proměnných a zkontroluje se počet vstupních parametrů. Dále proběhne kontrola korektnosti vstupních parametrů. Jelikož se jedná o stejný kód, který byl již použit v kapitole 4, odkazuji na jejich výpisy 36 a 38. Stejně zpracujeme i zbývající vstupní argumenty, viz výpis 39.
V další části ošetříme, zda-li není model filtru již otevřen, jak ukazuje výpis 56.

```
% kontrola, zda-li není model otevřen
if bdIsLoaded(model)
    % pokud je otevřen, model se zavře
    close_system(model,0)
end
```

Výpis 56: Kontrola otevření modelu

MDL soubor budeme tvořit jako textový soubor zápisem stringů pomoci příkazu $fprintf$, k čemuž bude využito dvou pracovních souborů 'temp1.tmp' a 'temp2.tmp'. Abychom s nimi mohli pracovat, je potřeba soubory otevřít pro zápis. K tomu slouží kód ve výpisu

```
fout1 = fopen('temp1.tmp','w');
fout2 = fopen('temp2.tmp','w');
```

Výpis 57: Otevření pracovních souborů pro zápis

Nyní můžeme začít s vytvářením vlastního souboru. Jako první do souboru vložíme údaje o modelu a začneme tvořit část **BlockParameterDefaults**. Jako první vložíme údaje bloku zpožd'ovače, viz výpis 58.

```
fprintf (fout1,'Model {\n');
fprintf (fout1,'  Name\t\t\t  "test"\n');
fprintf (fout1,'  Version\t\t  "8.5"\n');
fprintf (fout1,'  BlockParameterDefaults {\n');
fprintf (fout1,'    Block {\n');
fprintf (fout1,'      BlockType\t\t      Delay\n');
fprintf (fout1,'      DelayLengthSource\t    "Dialog"\n');
fprintf (fout1,'      DelayLength\t      "2"\n');
fprintf (fout1,'      DelayLengthUpperLimit "100"\n');
fprintf (fout1,'      InitialConditionSource  "Dialog"\n');
fprintf (fout1,'      InitialCondition \t      "0.0"\n');
fprintf (fout1,'      ExternalReset\t      "None"\n');
fprintf (fout1,'      ShowEnablePort\t    off\n');
fprintf (fout1,'      PreventDirectFeedthrough off\n');
fprintf (fout1,'      DiagnosticForOutOfRangeDelayLength "None"\n');
fprintf (fout1,'      RemoveProtectionDelayLength off\n');
fprintf (fout1,'      InputProcessing\t      "Elements as channels (sample based)"\n');
fprintf (fout1,'      UseCircularBuffer\t      off\n');
fprintf (fout1,'      SampleTime\t    "-1"\n');
fprintf (fout1,'      StateMustResolveToSignalObject off\n');
fprintf (fout1,'      CodeGenStateStorageClass "Auto"\n');
fprintf (fout1,'    }\n');
```

Výpis 58: Vložení dat zpožd'ovače

Následuje blok dat týkající se násobičky.

```
fprintf (fout1,'    Block {\n');
fprintf (fout1,'      BlockType\t\t      Gain\n');
fprintf (fout1,'      Gain\t\t      "1"\n');
fprintf (fout1,'      Multiplication \t      "Element-wise(K.*u)"\n');
fprintf (fout1,'      ParamMin\t\t      []\n');
fprintf (fout1,'      ParamMax\t\t      []\n');
```

```
fprintf (fout1,'        ParamDataTypeStr\t      "Inherit: Same as input"\n');
fprintf (fout1,'        OutMin\t\t          "[]"\n');
fprintf (fout1,'        OutMax\t\t          "[]"\n');
fprintf (fout1,'        OutDataTypeStr\t       "Inherit: Same as input"\n');
fprintf (fout1,'        LockScale\t\t         off\n');
fprintf (fout1,'        RndMeth\t\t        "Floor"\n');
fprintf (fout1,'        SaturateOnIntegerOverflow\ton\n');
fprintf (fout1,'        SampleTime\t\t       "-1"\n');
fprintf (fout1,'      }\n');
```

Výpis 59: Vložení dat násobičky

Posledním blokem dat **BlockParameterDefaults** k uložení je část určená pro sumu. To je zobrazeno ve výpisu 60.

```
fprintf (fout1,'      Block {\n');
fprintf (fout1,'        BlockType\t\t        Sum\n');
fprintf (fout1,'        IconShape\t\t        "rectangular"\n');
fprintf (fout1,'        Inputs\t\t         "++"\n');
fprintf (fout1,'        CollapseMode\t       "All dimensions"\n');
fprintf (fout1,'        CollapseDim\t        "1"\n');
fprintf (fout1,'        InputSameDT\t       on\n');
fprintf (fout1,'        AccumDataTypeStr\t      "Inherit: Inherit via internal rule"\n');
fprintf (fout1,'        OutMin\t\t          "[]"\n');
fprintf (fout1,'        OutMax\t\t          "[]"\n');
fprintf (fout1,'        OutDataTypeStr\t       "Inherit: Same as first input"\n');
fprintf (fout1,'        LockScale\t\t         off\n');
fprintf (fout1,'        RndMeth\t\t        "Floor"\n');
fprintf (fout1,'        SaturateOnIntegerOverflow\ton\n');
fprintf (fout1,'        RndMeth\t\t        "Floor"\n');
fprintf (fout1,'        SampleTime\t\t       "-1"\n');
fprintf (fout1,'      }\n   }\n');
```

Výpis 60: Vložení dat sumy

Další části MDL souboru je část **System**. Zde v úvodu této části uložíme velikost okna modelu po jeho otevření, viz výpis 61.

```
fprintf (fout1,'    System {\n');
fprintf (fout1,'      Name\t\t     "test"\n');
fprintf (fout1,'      Location\t\t      [20, 20, 1200, 700]\n');
```

Výpis 61: Vložení vlastností okna modelu

5.2 Automatizované generovaní nerekurzivní části filtru

V této podkapitole je popsán způsob tvorby nerekurzivní části filtru.
Jako první zjistíme řády obou části filtru a provedeme kontrolu, zda se na konci vektoru nevyskytuje nula. K tomu slouží kód ve výpisu 62.

```
% zjištění řádu filtru nerekurzivní části
orderB = length(numIn);
% zjištění řádu filtru rekurzivní části
orderA = length(denIn);

% eliminace nuly na konci vektoru
if numIn(orderB) == 0
    % pokud se nula vyskytuje, provede se dekrementace řádu filtru
    orderB = orderB - 1;
end
```

Výpis 62: Zjištění řádu obou části filtru

Následně vygenerujeme vstupy pro sumu dle řádu filtru, viz výpis 63.

```
bSumIn = '+|+';
if orderB > 2
    for i = 3 :orderB
        bSumIn = strcat(bSumIn, '|+');
    end
end
```

Výpis 63: Generování vstupů sumy

Každý vložený blok nebo propojení má přiřazen svůj ID. Ve výpisu 64 jsou nastaveny výchozí hodnoty ID. Po vložení nového bloku nebo propojky proměnné inkrementujeme.

```
% ID bloku
id = 1;
% ID propojky
id_l = 1;
```

Výpis 64: Generování vstupů sumy

Pokračujeme vložením bloku dat určeného pro generátor šumu, viz výpis 65.

```
fprintf(fout1,'    Block {\n');
fprintf(fout1,'      BlockType\t\t      Reference\n');
fprintf(fout1,'      Name\t\t      "Generator sumu"\n');
fprintf(fout1,'      SID\t\t      "%u"\n',id);
fprintf(fout1,'      Ports\t\t      [0, 1]\n');
fprintf(fout1,'      Position\t\t      [130, 45, 160, 75]\n');
fprintf(fout1,'      ZOrder\t\t      %u\n',id);
fprintf(fout1,'      LibraryVersion\t      "1.356"\n');
fprintf(fout1,'      SourceBlock\t      "simulink/Sources/Band-Limited White Noise"\n');
fprintf(fout1,'      SourceType\t      "Band-Limited White Noise."\n');
fprintf(fout1,'      ContentPreviewEnabled "1.356"\n');
fprintf(fout1,'      Cov\t\t      "[0.1]"\ n');
fprintf(fout1,'      Ts\t\t      "%.20e"\n',sampleTime);
fprintf(fout1,'      seed\t\t      "[23341]"\n');
fprintf(fout1,'      VectorParams1D\t      on\n');
fprintf(fout1,'    }\n');
id = id + 1;
```

Výpis 65: Vložení dat generátoru šumu

V dalším kroku provedeme umístění bloků násobiček s nastavením koeficientu a jejich propojení se sumou. Vzhledem k tomu, že se data pro propojky nacházejí až za údaji řešící umístění jednotlivých bloků, budeme tyto data ukládat do druhého pracovníího souboru 'temp2.tmp' a na konci funkce tyto soubory spojíme ve správném pořadí do výsledného souboru MDL. Ke vkládání bloků dat využijeme opět cyklus, jak ukazuje výpis 66.

```
for i = 1 :orderB, label = i - 1;
    % umístění násobiček s nastavením parametrů
    x_g = x+offsetX*3; y_g = y+offsetY; w_g = (x+offsetX*3)+w; h_g = (y+offsetY)+h;
     fprintf(fout1,'      Block {\n');
     fprintf(fout1,'        BlockType\t\t        Gain\n');
     fprintf(fout1,'        Name\t\t        "b%u"\n',label);
     fprintf(fout1,'        SID\t\t        "%u"\n',id);
     fprintf(fout1,'        Position\t\t        [%u, %u, %u, %u]\n',x_g, y_g, w_g, h_g);
     fprintf(fout1,'        ZOrder\t\t        %u\n',id);
     fprintf(fout1,'        Gain\t\t        "%.20f"\n',numIn(i));
     fprintf(fout1,'      }\n');
    id = id + 1;
    % propojení násobiček se sumou
     fprintf(fout2,'      Line {\n');
     fprintf(fout2,'        ZOrder\t\t        %u\n',id_l);
     fprintf(fout2,'        SrcBlock\t\t        "b%u"\n',label);
     fprintf(fout2,'        SrcPort\t\t        1\n');
     fprintf(fout2,'        DstBlock\t\t        "suma b"\n');
     fprintf(fout2,'        DstPort\t\t        %u\n',i);
     fprintf(fout2,'      }\n');
    id_l = id_l + 1;
```

Výpis 66: Vložení násobiček a propojení se sumou

Pokračujeme umístěním zpožď'ovačů a jejich konfigurací, viz výpis 67.

```
    if i > 1        % podmínka řeší umístění zpožďovače až po druhém cyklu
        x_d = (x+offsetX*2); y_d = ((y+offsetY)-55);
        w_d = (x+offsetX*2)+w; h_d = ((y+offsetY)-55)+h;
         fprintf(fout1,'      Block {\n');
         fprintf(fout1,'        BlockType\t\t        Delay\n');
         fprintf(fout1,'        Name\t\t        "Zpozdovac b%u"\n',label);
         fprintf(fout1,'        SID\t\t        "%u"\n',id);
         fprintf(fout1,'        Ports\t\t        [1, 1]\n');
         fprintf(fout1,'        Position\t\t        [%u, %u, %u, %u]\n',x_d, y_d, w_d, h_d);
         fprintf(fout1,'        ZOrder\t\t        %u\n',id);
         fprintf(fout1,'        BlockRotation\t        270\n');
         fprintf(fout1,'        BlockMirror\t        on\n');
         fprintf(fout1,'        ShowName\t\t        off\n');
         fprintf(fout1,'        InputPortMap\t        "u0"\n');
         fprintf(fout1,'        DelayLength\t        "1"\n');
         fprintf(fout1,'        SampleTime\t        "%.20e"\n',sampleTime);
         fprintf(fout1,'      }\n');
        id = id + 1;
    end
```

Výpis 67: Vložení zpožď'ovačů

Provedeme propojení 1. řádu filtru (násobičku a zpožď'ovač) s generátorem šumu, viz výpis 68.

```
if  i  == 1        % propojení s generátorem šumu se řeší pouze v 1. řádu
    fprintf (fout2,'      Line  {\n');
    fprintf (fout2,'         ZOrder\t\t        %u\n',id_l);
    fprintf (fout2,'         SrcBlock\t\t        "Generator sumu"\n');
    fprintf (fout2,'         SrcPort\t\t        1\n');
    fprintf (fout2,'         Points\t\t        [0,   0]\n');
    id_l  = id_l  + 1;
    fprintf (fout2,'         Branch { \n');
    fprintf (fout2,'\tZOrder\t\t\t        %u\n',id_l);
    fprintf (fout2,'\tDstBlock\t\t        "b%u"\n',label);
    fprintf (fout2,'\tDstPort\t\t\t        1\n');
    fprintf (fout2,'         }\n');
    id_l  = id_l  + 1;
    fprintf (fout2,'         Branch { \n');
    fprintf (fout2,'\tZOrder\t\t\t        %u\n',id_l);
    fprintf (fout2,'\tPoints\t\t\t        [80,   0]\n');
    fprintf (fout2,'\tDstBlock\t\t        "Zpozdovac b%u"\n',i);
    fprintf (fout2,'\tDstPort\t\t\t        1\n');
    fprintf (fout2,'         }\n');
    id_l  = id_l  + 1;
```

Výpis 68: Propojení 1. řádu filtru

Umístíme propojení generátoru šumu se spektrálním analyzérem. Musíme ale ošetřit stav, pokud se rekurzivní část nebude tvořit, jelikož propojení bude vedeno jinak, viz výpis 69.

```
    fprintf (fout2,'         Branch { \n');
    fprintf (fout2,'\tZOrder\t\t\t        %u\n',id_l);

   % umisteni propojení do spektrálního analyzéru
    if  ~((orderA == 1) && (denIn(1) == 1))
   % pokud bude rekurzivní část
    fprintf (fout2,'\tPoints\t\t\t        [145,  0;  0,  -35; 540, 0]\n');
   else
       % pokud nebude rekurzivní část
        fprintf (fout2,'\tPoints\t\t\t        [145,  0;  0,  -30; 240, 0]\n');
       end
    fprintf (fout2,'\tDstBlock\t\t        "Spektralni  analyzer"\n');
    fprintf (fout2,'\tDstPort\t\t\t        1\n');
    fprintf (fout2,'         }\n');
    fprintf (fout2,'      }\n');
    id_l  = id_l  + 1;
```

Výpis 69: Vložení propojky z generátoru šumu do spektrálního analyzéru

Dále vložíme propojky zpožď'ovačů a násobiček od druhého do předposledního řádu filtru, viz výpis 70.

```
    elseif  ((i > 1) && (i < orderB))
        fprintf(fout2,'      Line {\n');
        fprintf(fout2,'        ZOrder\t\t      %u\n',id_l);
        fprintf(fout2,'        SrcBlock\t\t      "Zpozdovac b%u"\n',label);
        fprintf(fout2,'        SrcPort\t\t      1\n');
        fprintf(fout2,'        Points\t\t      [0,  0]\n');
        id_l = id_l + 1;
        fprintf(fout2,'        Branch { \n');
        fprintf(fout2,'\tZOrder\t\t      %u\n',id_l);
        fprintf(fout2,'\tPoints\t\t      [0,  35]\n');
        fprintf(fout2,'\tDstBlock\t\t      "b%u"\n',label);
        fprintf(fout2,'\tDstPort\t\t      1\n');
        fprintf(fout2,'        }\n');
        id_l = id_l + 1;
        fprintf(fout2,'        Branch { \n');
        fprintf(fout2,'\tZOrder\t\t      %u\n',id_l);
        fprintf(fout2,'\tDstBlock\t\t      "Zpozdovac b%u"\n',i);
        fprintf(fout2,'\tDstPort\t\t      1\n');
        fprintf(fout2,'        }\n');
        fprintf(fout2,'      }\n');
        id_l = id_l + 1;
```

Výpis 70: Vložení propojky mezi zpožďovači a násobičkami - část 1

V posledním kroku cyklu umístíme propojení zpožďovače a násobičky posledního řádu filtru, viz výpis 71.

```
    else
        fprintf(fout2,'      Line {\n');
        fprintf(fout2,'        ZOrder\t\t      %u\n',id_l);
        fprintf(fout2,'        SrcBlock\t\t      "Zpozdovac b%u"\n',label);
        fprintf(fout2,'        SrcPort\t\t      1\n');
        fprintf(fout2,'        Points\t\t      [0,  35]\n');
        fprintf(fout2,'        DstBlock\t\t      "b%u"\n',label);
        fprintf(fout2,'        DstPort\t\t      1\n');
        fprintf(fout2,'      }\n');
        id_l = id_l + 1;
    end
    % na konci cyklu nastavíme offsetY k tvorbě další řady bloků
    offsetY = offsetY + 110;
end
```

Výpis 71: Vložení propojky mezi zpožďovači a násobičkami - část 2

Na závěr nerekurzivní části zbývá vložit sumu, viz výpis 72.

```
x_s = (x+offsetX*4); y_s = y+5; w_s = (x+offsetX*4)+w; h_s = h_g + 10;
fprintf(fout1,'      Block {\n');
fprintf(fout1,'        BlockType\t\t      Sum\n');
fprintf(fout1,'        Name\t\t      "suma b"\n');
fprintf(fout1,'        SID\t\t      "%u"\n',id);
fprintf(fout1,'        Ports\t\t      [%u, 1]\n',orderB);
fprintf(fout1,'        Position\t\t      [%u, %u, %u, %u]\n', x_s, y_s, w_s, h_s);
fprintf(fout1,'        ZOrder\t\t      "%u"\n',id);
```

```
fprintf (fout1,'        Inputs\t\t        "%s"\n',bSumIn);
fprintf (fout1,'      }\n');
id = id + 1;
```

Výpis 72: Vložení sumy

5.3 Automatizované generovaní rekurzivní části filtru

Následující podkapitola popíše tvorbu rekurzivní části filtru. U této části navíc přibude v kódu podmínka ošetřující stav, že pokud bude vstupní vektor obsahovat pouze jednu položku s hodnotou '1', nebude se tato část filtru vůbec generovat a bude nahrazena propojením.
V úvodu vložíme spektrální analyzér a nastavíme proměnnou **offsetY** na výchozí hodnotu, jak ukazuje výpis 73. Údaje o umístění budou vloženy později (výpis 77 a 85).

```
offsetY = 15;
% umístění spektrálního analyzéru
fprintf (fout1,'      Block {\n');
fprintf (fout1,'        BlockType\t\t        Reference\n');
fprintf (fout1,'        Name\t\t        "Spektralni analyzer"\n');
fprintf (fout1,'        SID\t\t        "%u"\n',id);
fprintf (fout1,'        Ports\t\t        [2]\n');
fprintf (fout1,'        ZOrder\t\t        %u\n',id);
fprintf (fout1,'        LibraryVersion\t        "1.43"\n');
fprintf (fout1,'        SourceBlock\t        "simulink_extras/Additional Sinks/Spectrum Analyzer"\n');
fprintf (fout1,'        SourceType\t        "Spectrum Analyzer"\n');
fprintf (fout1,'        ContentPreviewEnabled off\n');
fprintf (fout1,'        npts\t\t        "1024"\n');
fprintf (fout1,'        fftpts \t\t        "1024"\n');
fprintf (fout1,'        HowOften\t\t        "256"\n');
fprintf (fout1,'        SampleT\t\t        "%.20e"\n',sampleTime);
fprintf (fout1,'        HowOften\t\t        "256"\n');
id = id + 1;
```

Výpis 73: Vložení spektrálního analyzéru

Jak již bylo zmíněno v úvodu této podkapitoly, je potřeba ošetřit možost, že rekurzivní část filtru nebude potřeba generovat, pokud bude vstupní vektor obsahovat pouze jeden prvek s hodnotou '1'. Toho docílíme podmínkou ve výpisu 74, která bude platit pro další části kódu a je zakončena až ve výpisu 85. Pro přehlednost je tato část rozdělena na více části.

```
if ~((orderA == 1) && (denIn(1) == 1))
```

Výpis 74: Podmínka pro případ umístění propojení místo filtru

Stejně v jako nerekurzivní části, musíme elimimovat nulu na konci vektoru, viz výpis 75.

```
if denIn(orderA) == 0
    % pokud se nula vyskytuje, provede se dekrementace řádu filtru
     orderA = orderA - 1;
end
```

Výpis 75: Korekce řádu rekurzivní části filtru

Provedeme nastavení správného počtu vstupů sumy, viz výpis 76. Kvůli lepšímu zobrazení výsledného filtru je spodní vstup nastaven na menší vzdálenost.

```
if orderA > 2
        aSumIn='+|+';
        for i = 3 :orderA
            if i < orderA
            aSumIn = strcat(aSumIn, '|+');
            else
                aSumIn = strcat(aSumIn, '+');
            end
        end
    else aSumIn = '++';
end
```

Výpis 76: Generování vstupů sumy

Umístíme do modelu spektrální analyzér, pokud se bude tvořit rekurzivní část, viz výpis 77. Tato část je pokračováním výpisu 73 v závislosti na splnění podmínky ve výpisu 74.

```
fprintf(fout1,'        Position\t\t      [865, 47, 895, 78]\n');
fprintf(fout1,'    }\n');
```

Výpis 77: Vložení pozice spektrálního analyzéru

Pokračujeme vložením násobiček a zpožďovačů s nastavením parametrů. Budeme řešit opět s využitím cyklu.

```
for i = 2 :orderA, label = i - 1;
        x_g = (x+offsetX*6); y_g = (y+offsetY+50); w_g = (x+offsetX*6)+w; h_g = (y+offsetY+50)+h;
        x_d = (x+offsetX*5); y_d = (y+offsetY); w_d = (x+offsetX*5)+w; h_d = y+offsetY+h;

        % umístění násobiček s nastavením parametrů
        fprintf(fout1,'    Block {\n');
        fprintf(fout1,'      BlockType\t\t      Gain\n');
        fprintf(fout1,'      Name\t\t      "-a%u"\n',label);
        fprintf(fout1,'      SID\t\t      "%u"\n',id);
        fprintf(fout1,'      Position\t\t      [%u, %u, %u, %u]\n',x_g, y_g, w_g, h_g);
        fprintf(fout1,'      ZOrder\t\t      %u\n',id);
        fprintf(fout1,'      Gain\t\t      "%.20f"\n',-denIn(i));
        fprintf(fout1,'    }\n');
        id = id + 1;

        % umístění zpožďovačů s nastavením parametrů
        fprintf(fout1,'    Block {\n');
        fprintf(fout1,'      BlockType\t\t      Delay\n');
        fprintf(fout1,'      Name\t\t      "Zpozdovac a%u"\n',label);
        fprintf(fout1,'      SID\t\t      "%u"\n',id);
        fprintf(fout1,'      Ports\t\t      [1, 1]\n');
        fprintf(fout1,'      Position\t\t      [%u, %u, %u, %u]\n',x_d, y_d, w_d, h_d);
        fprintf(fout1,'      ZOrder\t\t      %u\n',id);
        fprintf(fout1,'      BlockRotation\t      270\n');
        fprintf(fout1,'      BlockMirror\t      on\n');
        fprintf(fout1,'      ShowName\t\t      off\n');
        fprintf(fout1,'      InputPortMap\t      "u0"\n');
```

```
fprintf (fout1,'        DelayLength\t       "1"\n');
fprintf (fout1,'        SampleTime\t      "%.20e"\n',sampleTime);
fprintf (fout1,'      }\n');
id = id + 1;
```

Výpis 78: Vložení spektrálního analyzéru

Propojíme jednotlivé násobičky se sumou, viz výpis 79.

```
fprintf (fout2,'    Line {\n');
fprintf (fout2,'      ZOrder\t\t        %u\n',id_l);
fprintf (fout2,'      SrcBlock\t\t       "-a%u"\n',label);
fprintf (fout2,'      SrcPort\t\t        1\n');
fprintf (fout2,'      DstBlock\t\t       "suma a"\n');
fprintf (fout2,'      DstPort\t\t        %u\n',i-1);
fprintf (fout2,'    }\n');
id_l = id_l + 1;
```

Výpis 79: Propojení násobiček se sumou

Vložíme propojky násobiček a zpožďovačů ve všech řádech filtru, mimo posledního, jak ukazuje výpis 80.

```
if (i < orderA)
    fprintf (fout2,'    Line {\n');
    fprintf (fout2,'      ZOrder\t\t        %u\n',id_l);
    fprintf (fout2,'      SrcBlock\t\t       "Zpozdovac a%u"\n',label);
    fprintf (fout2,'      SrcPort\t\t        1\n');
    fprintf (fout2,'      Points\t\t        [0,  0]\n');
    id_l = id_l + 1;
    fprintf (fout2,'      Branch { \n');
    fprintf (fout2,'\tZOrder\t\t        %u\n',id_l);
    fprintf (fout2,'\tPoints\t\t        [0,  30]\n');
    fprintf (fout2,'\tDstBlock\t\t       "-a%u"\n',label);
    fprintf (fout2,'\tDstPort\t\t        1\n');
    fprintf (fout2,'      }\n');
    id_l = id_l + 1;
    fprintf (fout2,'      Branch { \n');
    fprintf (fout2,'\tZOrder\t\t        %u\n',id_l);
    fprintf (fout2,'\tDstBlock\t\t       "Zpozdovac a%u"\n',i);
    fprintf (fout2,'\tDstPort\t\t        1\n');
    fprintf (fout2,'      }\n');
    fprintf (fout2,'    }\n');
    id_l = id_l + 1;
```

Výpis 80: Vložení propojky mezi zpožďovači a násobičkami - část 1

A následně propojíme násobičku se zpožďovačem v posledním řádu filtru a zakončíme cyklus, viz výpis 81.

```
else
        fprintf (fout2,'    Line {\n');
        fprintf (fout2,'      ZOrder\t\t        %u\n',id_l);
        fprintf (fout2,'      SrcBlock\t\t       "Zpozdovac a%u"\n',label);
        fprintf (fout2,'      SrcPort\t\t        1\n');
```

```
            fprintf(fout2,'        Points\t\t      [0, 30]\n');
            fprintf(fout2,'        DstBlock\t\t      "a%u"\n',label);
            fprintf(fout2,'        DstPort\t\t      1\n');
            fprintf(fout2,'      }\n');
            id_l = id_l + 1;
        end
    % na konci cyklu nastavíme offsetY k tvorbě další řady bloků
    offsetY = offsetY + 110;
    end
```

Výpis 81: Vložení propojky mezi zpožďovači a násobičkami - část 2

Následuje vložení propojení ze sumy do spektrálního analyzéru a prvního zpožďovače, viz výpis 82.

```
fprintf(fout2,'    Line {\n');
fprintf(fout2,'      ZOrder\t\t      %u\n',id_l);
fprintf(fout2,'      SrcBlock\t\t      "suma a"\n');
fprintf(fout2,'      SrcPort\t\t      1\n');
fprintf(fout2,'      Points\t\t      [0, 0]\n');
id_l = id_l + 1;
fprintf(fout2,'      Branch { \n');
fprintf(fout2,'\tZOrder\t\t      %u\n',id_l);
fprintf(fout2,'\tPoints\t\t      [0, %d]\n',-(orderA-1)*55-55);
fprintf(fout2,'\tDstBlock\t\t      "Zpozdovac a1"\n');
fprintf(fout2,'\tDstPort\t\t      1\n');
fprintf(fout2,'      }\n');
id_l = id_l + 1;
fprintf(fout2,'      Branch { \n');
fprintf(fout2,'\tZOrder\t\t      %u\n',id_l);
fprintf(fout2,'\tPoints\t\t      [40, 0; 0, %d]\n',-(orderA-1)*55-15);
fprintf(fout2,'\tDstBlock\t\t      "Spektralni analyzer"\n');
fprintf(fout2,'\tDstPort\t\t      2\n');
fprintf(fout2,'      }\n');
fprintf(fout2,'    }\n');
id_l = id_l + 1;
```

Výpis 82: Propojení ze sumy do spektrálního analyzéru a prvního zpožďovače

V dalším kroku vložíme propojení obou sum, viz výpis 83.

```
fprintf(fout2,'    Line {\n');
fprintf(fout2,'      ZOrder\t\t      %u\n',id_l);
fprintf(fout2,'      SrcBlock\t\t      "suma b"\n');
fprintf(fout2,'      SrcPort\t\t      1\n');
fprintf(fout2,'      Points\t\t      [60, 0; 0, %u]\n',(orderA-1)*55-5);
fprintf(fout2,'      DstBlock\t\t      "suma a"\n');
fprintf(fout2,'      DstPort\t\t      %u\n',orderA);
fprintf(fout2,'    }\n');
```

Výpis 83: Vložení propojky mezi sumami

Dále umístíme sumu, viz výpis 84.

```
x_s = (x+offsetX*7); y_s = y+50; w_s = (x+offsetX*7)+w; h_s = h_g + 70;
 fprintf (fout1,'      Block {\n');
 fprintf (fout1,'        BlockType\t\t       Sum\n');
 fprintf (fout1,'        Name\t\t       "suma a"\n');
 fprintf (fout1,'        SID\t\t       "%u"\n',id);
 fprintf (fout1,'        Ports\t\t       [%u, 1]\n',orderA);
 fprintf (fout1,'        Position\t\t       [%u, %u, %u, %u]\n', x_s, y_s, w_s, h_s);
 fprintf (fout1,'        ZOrder\t\t       %u\n',id);
 fprintf (fout1,'        Inputs\t\t       "%s"\n',aSumIn);
 fprintf (fout1,'      }\n');
```

Výpis 84: Vložení sumy

Pokud nebudeme tvořit rekurzivní část, vložíme spektrální analyzér blíže k 'sumě b' a tyto dva bloky propojíme, viz výpis 85. Tato část je pokračováním výpisu 73 v závislosti na splnění podmínky ve výpisu 74.

```
else
    % nastavení umístění spektr. analyzéru, pokud nebude rekurzivní část
     fprintf (fout1,'        Position\t\t       [565, 47, 595, 78]\n');
     fprintf (fout1,'      }\n');
    % propojení sumy b do spektr. analyzéru, pokud nebude rekurz. část
     fprintf (fout2,'      Line {\n');
     fprintf (fout2,'        ZOrder\t\t       %u\n',id_I);
     fprintf (fout2,'        SrcBlock\t\t       "suma b"\n');
     fprintf (fout2,'        SrcPort\t\t       1\n');
     fprintf (fout2,'        Points\t\t       [40, 0; 0, -265]\n');
     fprintf (fout2,'        DstBlock\t\t       "Spektralni analyzer"\n');
     fprintf (fout2,'        DstPort\t\t       2\n');
     fprintf (fout2,'      }\n');
end
 fprintf (fout2,'    }\n}');
```

Výpis 85: Vložení pozice spektrálního analyzéru a propojení se sumou

5.4 Závěrečná část funkce

V této části funkce zpracujeme pracovní soubory, ze kterých vytvoříme finalní soubor MDL a uložíme obrázek se zobrazením modelu do souboru PNG, případně spustíme simulaci.
Abychom mohli s pracovními soubory dále pracovat, musíme provést jejich uzavření. To provedem následujícími příkazy ve výpisu 86.

```
fclose(fout1);
fclose(fout2);
```

Výpis 86: Uzavření pracovních souborů

Následně vytvoříme finální verzi MDL souboru, viz výpis 88. Jsou zde ošetřeno použití v systémech Windows a Linux, jelikož se zde využívají systémové příkazy, které se v závoslosti na operačním systému liší.

```
if ispc
    % v systému Windows
    system(['copy temp1.tmp+temp2.tmp ',file])
    system('del temp1.tmp temp2.tmp')
    elseif isunix
        % v systému Linux
        system(['cat temp1.tmp temp2.tmp > ',file])
        system('rm temp1.tmp temp2.tmp')
end
```

Výpis 87: Vytvoření finálního MDL souboru

Na závěr otevřeme okno s modelem, spustíme simulaci, je-li vyžadováno a uložíme zobrazení modelu do souboru PNG.

```
% otevření okna modelu
open_system(model);
% spuštění simulace, pokud bylo zadáno argumentem
if simOn == 1; sim(model); end
% uložení zobrazení modelu do souboru png
print ([ '-s', model], '-dpng', ['model_', model, '.png']) ;
% zakončení funkce
end
```

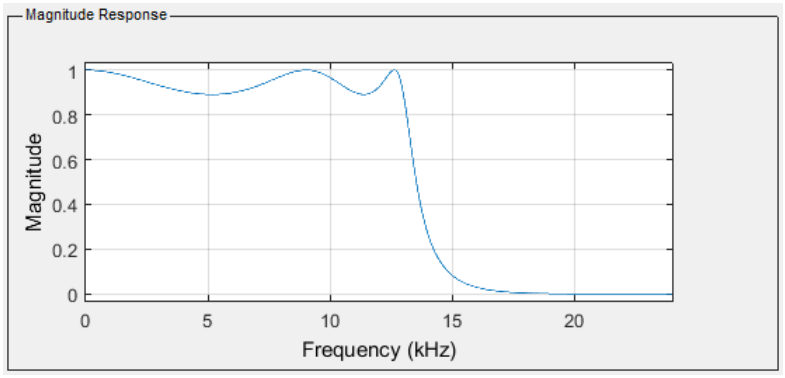
Výpis 88: Závěrečná část funkce

5.5 Ukázka použití funkce

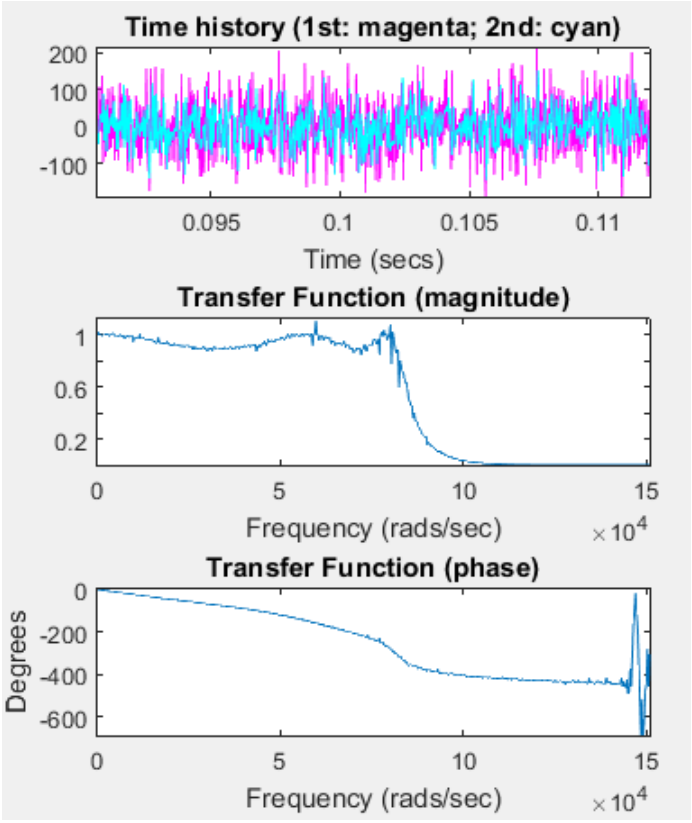
K vygenerování MDL souboru využijeme stejného zadání, jako bylo použito v kapitole 4.5, abychom mohli srovnat shodnost obou způsobu tvorby filtrů.
Oba vektory s koeficienty, stejně jako v předešlém případě, vyexportujeme na Workspace. Vektor pro nerekurzivní část je pojmenován jako 'Num' a pro rekurzivní jako 'Den'. Vzorkovací frekvenci máme 48000Hz. Nyní můžeme začít s generováním MDL souboru. Do 'Command window' tedy zadáme následující příkaz:

postav_mdl(Num,Den,48000,1)

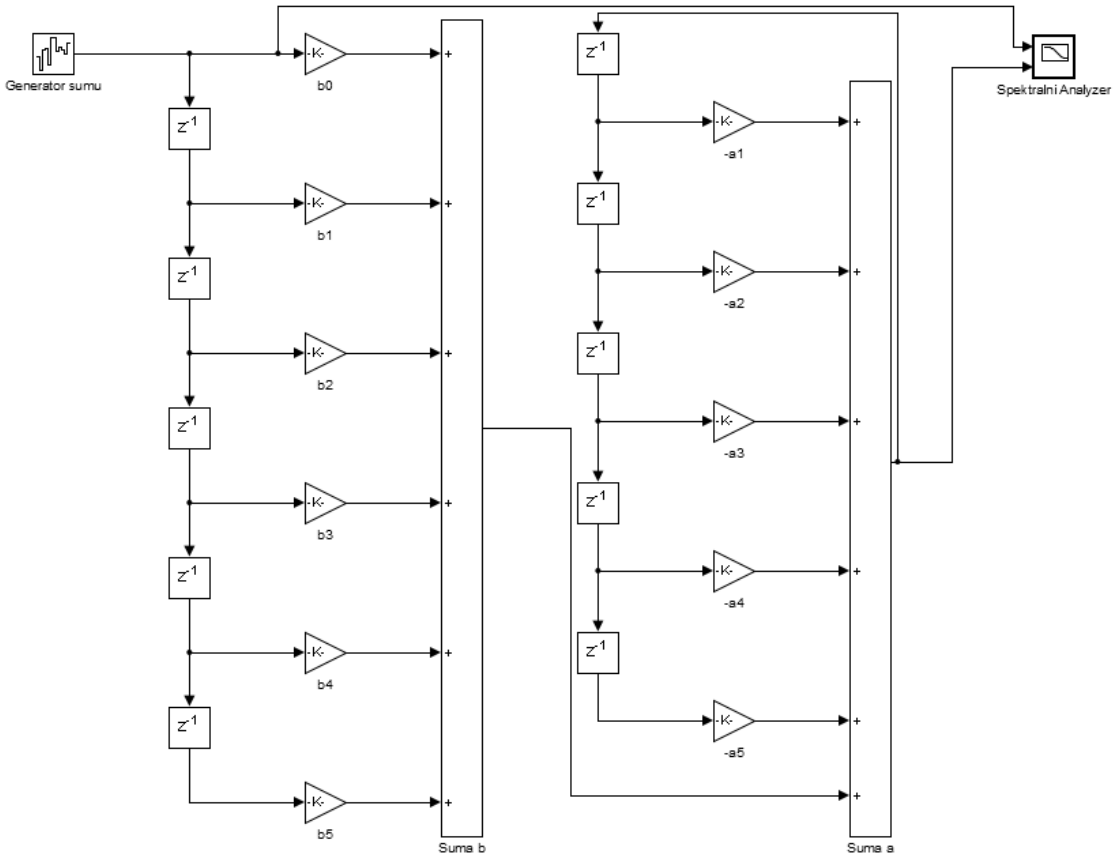
Tímto se nám automaticky vygeneruje MDL soubor modelu digitálního IIR filtru 5. řádu viz obrázek 23 a zobrazí se jeho přenosová charakteristika v otevřeném okně spektrálního analyzéru viz obrázek 22. Pro porovnání je na obrázku 21 zobrazena frekvenční charakteristika vygenerovaná FDATOOLem. Je patrné, že obě charakteristiky jsou totožné.



Obrázek 21: Frekvenční přenosové charakteristika vygenerované FDATOOLem



Obrázek 22: Frekvenční přenosové charakteristika zobrazená spektrálním analyzérem



Obrázek 23: Otevřený vygenerovaný MDL soubor digitálního IIR filtru 5. řádu

Závěr

Ve své bakalářské práci jsem se v teoretickém úvodu věnoval popisu struktury souboru MDL. V praktické části byla detailně popsána tvorba matlabovské funkce pro automatizované generování modelu. Pro demonstraci byl zvolen digitální IIR filtr v přímé nekanonické formě, jehož model byl generován na základě přechozího výpočtu v prostředí MATLAB. Tvorba vlastního modelu digitálního filtru byla předvedena jak s využitím matlabovských příkazů pro vkládání bloků a propojení, tak i přímou tvorbou souboru MDL.
V závěru práce jsem provedl porovnání vygenerovaných modelů filtrů, vytvořených oběma způsoby a jejich simulace. Z výsledků simulací je patrné, že užití obou variant ke tvorbě modelů nemá na vlastnosti vytvořených digitálních filtrů žádný vliv a provedené simulace jsou totožné.

Literatura

[1] MATLAB. *Humusoft s.r.o.*, [online]. ©1991 - 2016 cit. [2016-01-23]. Dostupné z: http://www.humusoft.cz/matlab

[2] Model File Format. *ROHAN Academic Computing*, [online]. 2015 cit. [2016-01-23]. Dostupné z: http://www-rohan.sdsu.edu/doc/matlab/toolbox/simulink/slref/model_file_fmt.html

[3] Building Models with MATLAB Code. *Mathworks*, [online]. 2010 cit. [2016-01-23]. Dostupné z http://blogs.mathworks.com/simulink/2010/01/21/building-models-with-matlab-code

[4] Building Simulink Models using MATLAB Code. *Goddard Consulting*, [online]. ©2003 - 2015 cit. [2016-01-23]. Dostupné z: http://www.goddardconsulting.ca/simulink-creating-using-matlab-code.html

[5] DAVÍDEK, Vratislav, Miloš LAIPERT a Miroslav VLČEK. *Analogové a číslicové filtry.* Vydání 1. Praha: Vydavatelství ČVUT, 2000. ISBN 80-01-02178-5.

[6] XUE, Dingyü, YangQuan CHEN. *System Simulation Techniques with MATLAB and Simulink.* Vydání 1. Chichester, United Kingdom: Vydavatelství John Wiley & Sons, ©2014.

[7] KARBAN, Pavel. *Výpočty a simulace v programech Matlab a Simulink.* Vydání 1. Brno: Computer Press, 2006. ISBN 80-251-1301-9.

[8] KOZÚBEK, Tomáš. *Programování v Matlabu*, [online]. Ostrava: VŠB-TUO, 2009 [cit. 2016-03-27]. Dostupné z: homel.vsb.cz/~dom033/predmety/NMM/01/lekce1.pdf

Seznam příloh

Příloha na CD:

block.mdl	model obsahující jeden blok
twoblocks.mdl	model obsahující dva bloky s propojením
digfiltr.m	zdrojový kód jednoduchého dig. filtru
digfiltr_1_r.m	zdrojový kód dig. filtru 1. řádu
digfiltr_2_r.m	zdrojový kód dig. filtru 2. řádu
postav_filtr.m	zdrojový kód funkce k automatizované tvorbě dig. filtru libovolného řádu příkazy MATLABu
postav_mdl.m	zdrojový kód funkce k automatizované tvorbě dig. filtru libovolného řádu přímou tvorbou MDL souboru
workspace.mat	vyexportované vstupní vektory z ukázky generování filtru v kapitolách 4.5 a 5.5 pro funkce postav_filtr a postav_mdl
zadani.jpg	zadání bakalářské práce
práce.pdf	text bakalářské práce